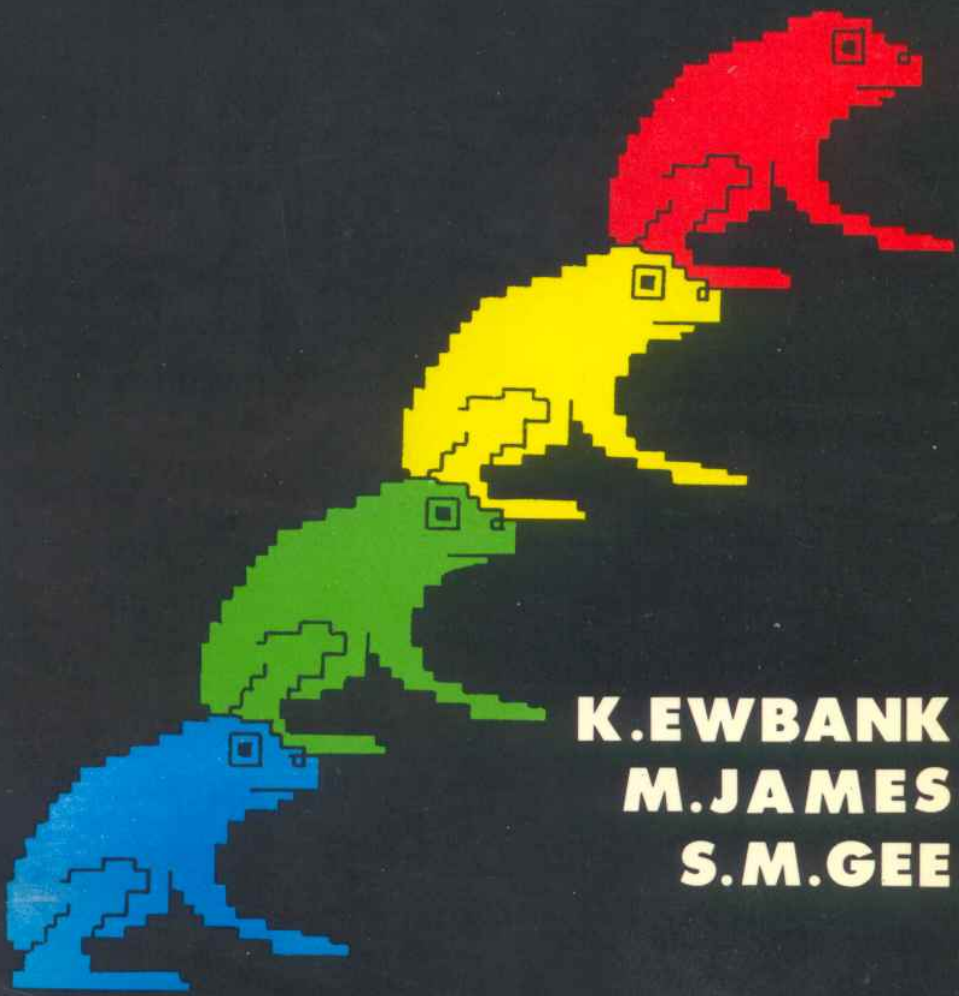
SPECTRUM

diseño de juegos



K.EWBANK
M.JAMES
S.M.GEE

Kay Ewbank, Mike James
y S. M. Gee

SPECTRUM
Diseño de juegos

Traducido por
Aula de Informática Aplicada

Ediciones DIAZ DE SANTOS, S. A. - Madrid-Barcelona

Copyright © Kay Ewbank, Mike James y S. M. Gee, 1984

Queda prohibida la reproducción total o parcial de la presente obra, bajo cualquiera de sus formas, gráfica o audiovisual, sin la autorización previa o escrita del editor, excepto citas en revistas, diarios o libros, siempre que se mencione la procedencia de las mismas.

ISBN: 0-246-12515-2 (edición en lengua inglesa).
ISBN: 84-86251-30-3 (edición en lengua española).

Título original: «Spectrum Gamesmaster»

Traducción, supervisión y adaptación de:
Aula de Informática Aplicada, S.A.

Coordinación: Manuel Garrido Pérez.

Diseño de portada: Francisco Bussalleu

Edita: DIAZ DE SANTOS, S. A.
C/ Juan Bravo, 3A
28006-MADRID

R: 62.222

Depósito Legal: M-42.399-1985
Primera Edición
Imprime: Imprenta CALERO - Madrid.

SPECTRUM
Diseño de juegos

U46000258602

U46000258602

Contenido

Prólogo 9

1. El aprendiz de brujo 11

Cómo escribir programas 12
La necesidad de un método 13
Refinamiento paso a paso y subrutinas 14
Estructura y estilo 16
Convenios relativos a los listados 17
Técnicas de los juegos 18

2. La colina de las hormigas 19

Diseño del juego 19
La técnica de la animación 22
El programa principal 24
Subrutinas 1000 y 8000 24
Subrutina 2000 28
Subrutina 4000 - El corazón del juego 30
Subrutina 6000 - Sección del fin del juego 33
Mejorando el juego 37
Sugerencias para posteriores trabajos 39
La versión final - Listado completo 39

3. El salto de la rana 49

Diseño del juego 49
Botando y cayendo 50
El programa principal 52
Subrutinas 1000 y 6000 53
Subrutina 2000 57
Subrutina 5000 57
Subrutinas 3000 y 4000 58
Subrutinas 4500, 8000 y 7000 60
Subrutina 9000 61
Evaluación y mejoras 62
La versión final - Listado completo 63

4. **Haciendo la Rana** 71

Animación por deslizamiento de pantalla 71
Diseño del juego 72
Código máquina 72
Programa principal y subrutina 7000 - La rutina del Título 77
Subrutina 1000 - Sección de inicialización 80
Subrutina 3000 y subrutinas asociadas 82
Subrutinas 6000 y 6500 86
Subrutinas 6600 a 8000 87
Evaluación y mejoras 88
La versión final - Listado completo 89

5. **La Serpiente** 97

Animación de serpientes 97
La serpiente dirigida - «Colas» 99
El diseño del juego 101
El programa principal 101
Subrutina 1000 102
Encuadre del Título 105
Subrutinas 2000 (dibujar escena), 2600 (dibujar serpiente) y 2500 (dibujar comida) 107
Subrutina 3000 y subrutinas asociadas 109
La subrutina 5000 y subrutinas asociadas 112
Subrutina 7000 113
Conclusión - Cómo jugar 116
La versión final - Listado completo 116

6. **Los renacuajos** 127

El diseño del juego 127
Puesta en escena 128
Haciendo una escapada 130
Cazando ranas 131
La versión final - Listado completo 133

7. **Serpientes y Escalerillas** 145

Diseño del juego 145
El programa principal 147
Subrutinas 1000 y 8000 147
Impresión del Tablero - Subrutina 2500 y subrutinas asociadas 151
Subrutina 3700 154

Hombre en movimiento - Subrutina 5000 y subrutinas asociadas 154
La sección final del juego 160
Evaluación 160
La versión final - Listado completo 161
Serpientes y Escaleras de acción 170

8. Convertido en programador experto 175

Juegos y solución de problemas 175
Añadiendo los toques finales 177
Posibles juegos 179
El camino hacia adelante 180

Prólogo

Programar bien, como cualquier otra habilidad, requiere práctica. Pero no se trata de resolver varios ejercicios; en algún momento hay que empezar a desarrollar ideas propias. Muchos principiantes encontrarán esto poco alentador, y preferirán ponerse en manos de algún experto. Este libro intenta ofrecer la misma ayuda que, un programador con experiencia, ofrece a una persona inexperta. El libro nos conducirá a través de todas las etapas del diseño de un programa, señalando los problemas y ofreciendo soluciones.

Todos los programas analizados son juegos gráficos animados, y algunos de ellos son los juegos clásicos «de marcianos». El lector puede pensar que programar dichos juegos está por encima de sus posibilidades, pero con la ayuda de este libro pronto será capaz de escribir juegos con el mismo nivel de dificultad.

Los programas han sido escritos para un Spectrum de 48 K. y, por tanto, para modificarlos o ampliarlos se recomienda un ordenador de 48 K. Cuatro de los juegos —«La Colina de las Hormigas», «El salto de la rana», «Serpientes y Escalerillas» y «Serpientes y Escaleras de acción»— se pueden ejecutar en una máquina de 16 K, pero los otros dos —«La Serpiente» y «Los Renacuajos»— no. «Haciendo la rana» se puede modificar para un 16 K, cambiando las direcciones de la rutina en Código Máquina.

Los programas están construídos como una serie de módulos, y le sugerimos que los teclee de uno en uno, tal y como se presentan. Así, no sólo se reduce el trabajo, sino que aumentan las posibilidades de entender el programa hasta en los más finos detalles. Una vez tecleados los juegos, esperamos que le resulten divertidos, que no tenga miedo de «Jugar» con ellos para mejorarlos o modificarlos, y que tenga también el valor suficiente para escribir sus propios programas.

Nuestro agradecimiento a Richard Miles y Prue Harrison de la Editorial Granada, sin cuya ayuda este libro no habría sido posible.

Kay Ewbank, Mike James and S.M.Gee

1. El aprendiz de brujo

Este libro es bastante diferente de cualquiera que haya leído hasta ahora sobre programación de computadoras. No se trata simplemente de una colección de juegos divertidos, ni tampoco es un libro más para «Aprender a Programar». El «Spectrum, diseño de juegos» une la diversión de crear y jugar, con la oportunidad de adquirir unos conocimientos que sólo se consiguen con la experiencia práctica.

Cada vez hay más gente que conoce un lenguaje de programación, pero conocer un lenguaje es sólo el principio, y luego hay que aprender a usarlo.

La mayor parte de la gente encuentra que, después de aprender BASIC, todavía tiene problemas en escribir programas que hagan «algo» de verdad.

Las dificultades varían desde no ser capaz de hacer un principio de programa, hasta no ser capaz de producir un programa acabado y útil, pasando por no ser capaz de resolver problemas que aparecen durante el desarrollo del programa. En otras palabras, los problemas pueden aparecer al principio, al final, o en mitad del desarrollo del programa, y, por supuesto, en cualquiera combinación de los tres. El caso es que, al menos al principio, es necesario un gran despliegue de energía para escribir un programa largo.

Los programas cortos se pueden escribir, probar y depurar en una parte, pero un programa largo puede llevarnos meses de trabajo. Las dificultades que aparecen trabajando con un gran número de líneas de BASIC, durante un largo período de tiempo, son nuevas, y bastante diferentes de lo que cualquier «Introducción al BASIC» nos dice. Cuando hay que enfrentarse a problemas reales con el BASIC, no es suficiente conocer, por ejemplo, qué hace la sentencia IF; es esencial conocer cómo utilizarla en combinación con el resto del BASIC, para conseguir una solución en la forma de un programa, que no sólo funcione, sino que sea «un placer» utilizarlo.

El «Spectrum, diseño de juegos» pretende tratar los problemas de escribir programas largos, explicando cómo fueron escritos varios programas largos de juegos.

La razón del énfasis en los juegos, es simplemente, que su atractivo es (casi) universal y que, además, contienen la mayor parte de los proble-

mas que se pueden encontrar en otras áreas de aplicación. ¿Dónde, si no, se podrían combinar técnicas de gráficos, efectos de sonido, expresiones aritméticas complicadas, números aleatorios, e incluso lógica Booleana en un único programa?. Una segunda razón es que los juegos son divertidos y, si se dispone de un ordenador, puede resultar interesante.

Si usted sólo quiere utilizar el libro como una colección de juegos, entonces ¡adelante!, vaya directamente a los listados completos al final de cada capítulo y tecléelos, pero disfrutará mucho más del juego si sigue todo el capítulo y teclea el programa por partes.

Cómo escribir programas

No se puede negar el hecho de que la mejor manera de aprender a escribir programas sustanciales, es trabajar con un programador experimentado. En este sentido, la programación no es diferente de cualquier otra materia, se necesita empezar como aprendiz antes de llegar a ser un maestro. El problema es que, en el momento actual, es difícil encontrar un maestro en programación. Si pudiese encontrar alguno, no desperdicie la oportunidad de aprender de él directamente, ya que es la mejor forma, pero, si no puede, el «Spectrum, diseño de juegos» le resultará muy útil.

Cada capítulo comienza con una descripción del juego que se va a escribir, señalando algunas de las posibles dificultades, y sugiriendo métodos que probablemente le resultarán útiles. Después de leer esta introducción, tendrá la misma información que un programador experto tendría antes de empezar a resolver los problemas inesperados, que surgen durante el desarrollo del programa.

De acuerdo con esto, cada programa es presentado por secciones. Cada sección del programa realiza una única acción dentro del mismo, y se explica en el texto que la acompaña. Le sugerimos que teclee el programa por secciones, ya que no sólo es más cómodo, sino que además le da la oportunidad de entender mejor cada una de ellas.

Hacia el final de cada capítulo encontrará una discusión acerca de lo que se ha realizado en el programa, y cómo mejorarse éste. La versión final de cada programa se puede encontrar listada con las modificaciones necesarias para hacer más divertidos los juegos. Los listados completos de los programas se pueden utilizar para comprobar que ha tecleado todo correctamente, incluyendo las modificaciones.

Es necesario advertir que todos los programas de este libro han sido comprobados, y, para evitar errores tipográficos, los listados son de impresora.

Las explicaciones de cómo funcionan los programas y de por qué las

cosas se hicieron de una determinada manera, pueden ayudarle a entender el proceso de escritura de programas largos, pero, por razones de extensión, no es posible mostrar todos los pasos de los mismos. A pesar de todo, los programas no han sido «pulidos», por ejemplo, no se han renumerado para conservar la evidencia de los problemas que pueden surgir a lo largo de la confección de los mismos.

Hay una tendencia a publicar los programas perfectamente pulidos, hasta el punto de que no quede ni rastro de los problemas que el programador tuvo, o bien dejarlos tan enredados, que pueden funcionar, pero imposibles de entender o modificar. En este libro, los programas no han sido «pulidos» en absoluto, y por la extraña numeración de las líneas se pueden encontrar algunas que han sido insertadas en una segunda etapa. Por ejemplo, si encuentra la secuencia de líneas 10, 20, 25, 30, 40..... es que el programador tuvo que retroceder, e insertar la línea 25, para tener en cuenta algo inesperado. Por otro lado, todos los programas han sido escritos con un método de programación dirigido a producir programas fáciles de entender.

Una vez tecleada cada sección de programa, sería interesante comprobar si se ha entendido, ejecutándolo por partes. Por supuesto que, a menos que se añadan líneas auxiliares, se obtendrán mensajes de error del programa parcial. Esta estrategia puede resultar muy útil para aumentar la confianza en lo que se está haciendo, y pensar más a fondo en el programa. Este es un libro esencialmente práctico, y por tanto, los programas son muy flexibles, y se pueden modificar para experimentar, bien mientras se están escribiendo, o bien una vez ya escritos. Incluso el decir que un programa está completo es algo relativo; en los ambientes Informáticos se dice que «no hay ningún programa 'terminado', simplemente se ha escrito hasta que ha resultado 'útil'». Para que resulte más fácil modificar y añadir líneas a los programas, al final de la mayor parte de los capítulos, se dan sugerencias, mejoras, e incluso nuevas versiones del juego.

Aunque este libro es eminentemente práctico, se necesita un mínimo de conocimientos teóricos, y por tanto, al final de este capítulo se describen algunas ideas que subyacen en los programas.

Si ya está familiarizado con estas ideas, tranquilamente puede pasar al siguiente capítulo, pero refrescarlas siempre es útil.

La necesidad de un método

Como ya se ha mencionado anteriormente, escribir programas cortos es muy diferente de escribir programas largos. Un programa corto se puede escribir «de golpe», y en la mayor parte de los casos puede estar «guardado» en la cabeza del programador. Sin embargo, un programa largo lleva demasiado tiempo como para escribirlo de una vez, y tiene

tantas líneas de BASIC, que salvo raras excepciones el programador no puede recordarlo. Se pueden escribir programas largos del mismo modo que los cortos, pero es un trabajo mucho más duro, y es por esta razón por lo que muchos programadores abandonan los proyectos muy amplios, y producen muy pocos programas realmente «acabados».

Incluso si se ha gastado el tiempo y esfuerzo necesario para producir un programa largo, el resultado final es generalmente tan caótico, que caulquier modificación, mejora, o intento de búsqueda de errores resulta imposible.

Esta situación resulta muy penosa, porque representa muchas horas de esfuerzo perdido. La verdad es que, si se utiliza un método de programar para organizar este esfuerzo, entonces escribir programas largos no será más difícil que escribir un programa corto.

A menudo, la sugestión de estar utilizando un método «llena de temor los corazones de los programadores amantes de la libertad». Este deseo de ser poco estricto al programar, es especialmente comprensible cuando se programa como diversión. Después de todo, ¿Cuál es el punto en que una diversión reglamentada se vuelve actividad rutinaria y parece un trabajo? La respuesta es que un método de programación no es una «receta» para reglamentar, es simplemente el «arte» de la programación.

Refinamiento paso a paso y subrutinas

El refinamiento paso a paso o progresivo no es sólo es un método de programación, es una estrategia general para resolver problemas de cualquier clase, y, por tanto, es muy valioso. Para entendernos, está basado en la antigua idea de «divide y vencerás».

Un programa largo se puede dividir en una colección de programas más pequeños. A su vez, los programas más pequeños se pueden volver a dividir, y así el resultado es suficientemente pequeño como para ser tratado de una vez. Dicho de esta manera, el refinamiento progresivo parece algo obvio, pero todavía queda el problema de cómo dividir el programa largo.

Esto es algo que se irá demostrando con cada uno de los programas del libro. Sin embargo, se puede decir algo más acerca de la teoría del refinamiento por pasos. Supongamos que queremos escribir un programa para jugar al ajedrez. El mayor problema para muchos programadores sería el de escribir el comienzo. Si pensamos un momento, rápidamente veremos

que cualquier programa para jugar al ajedrez, antes que nada debe dibujar el tablero, inicializar todas las piezas y luego ofrecer al jugador humano la posibilidad de mover, después hacer un movimiento y seguir así hasta el jaque-mate. O, lo que es lo mismo escrito en BASIC:

```
10 GOSUB 1000: REM dibujo del tablero
20 REM bucle de vuelta
30 GOSUB 2000: REM movimiento del hombre
40 GOSUB 3000: REM movimiento de la máquina
50 IF NOT JAQUEMATE THEN GOTO 20
60 STOP
```

Esto no es más que el primer paso de un buen programa para jugar al ajedrez. En otras palabras, no es más que el primer paso del método de refinamiento.

El siguiente paso consiste en escribir las subrutinas utilizadas en la parte del programa anterior (llamado generalmente programa principal).

Cada una de estas subrutinas se puede escribir dividiéndola a su vez en otras, así hasta que el programa queda escrito.

Es evidente que, utilizando este método, es más fácil escribir el comienzo del programa, simplemente haciendo llamadas a subrutinas que aún no se han escrito. También de este modo evitamos el tener que resolver problemas difíciles hasta que realmente necesiten ser resueltos. Por ejemplo, en el programa de ajedrez no tenemos que resolver el problema del turno de movimiento de la máquina hasta un paso bastante lejano del refinamiento.

Otra ventaja del uso de subrutinas como parte del refinamiento, es que así el programa resultante es muy fácil de entender y modificar.

Por ejemplo, observando el programa anterior, cualquiera puede deducir que el jugador humano siempre mueve el primero. Esto se podría modificar muy fácilmente, simplemente intercambiando las líneas 30 y 40. Imaginemos lo difícil que sería hacer este cambio si las partes del programa encargadas de los movimientos no estuvieran separadas del resto en forma de subrutinas.

Utilizando este método se consiguen tantas ventajas que es imposible enumerarlas todas. Muchas de ellas se harán evidentes en los ejemplos de los capítulos siguientes, pero, si se desea conocer algo más acerca de métodos de programación, puede consultar «Expert Guide to the Spectrum» de Mike James (Granada, 1984).

Estructura y estilo

El refinamiento paso a paso es un método que ayuda a escribir programas sin más que «partirlos», haciendo llamadas a varias subrutinas. A pesar de esto, llegará un momento en que una subrutina deberá hacer algo más que llamar a otras subrutinas.

El refinamiento no nos da ninguna pista acerca de cómo hemos de utilizar el BASIC para conseguir un determinado resultado. El método de programación que realmente sirve para esto se llama programación estructurada.

La programación estructurada es esencialmente una técnica para escribir programas claros, sin más que llevar el flujo de control del programa evitando los desórdenes.

Por ejemplo, al utilizar la sentencia GOTO se puede saltar a cualquier parte del programa, pero, si se utiliza GOTO libremente, se comprueba que los programas pueden resultar muy difíciles de entender y depurar. Es muy difícil seguir lo que sucede en un programa que salta de un sitio a otro, y, si no se puede seguir el orden que llevan las diferentes sentencias, es muy posible que se cometan errores . La programación estructurada nos asegura que es siempre fácil seguir el flujo de control, ya que al programador sólo se le permiten unas pocas formas de alterarlo.

Las sentencias más usuales para alterar el flujo de control, son IF y los bucles FOR, y además cualquier programa se puede escribir utilizando únicamente estas dos sentencias. En otras palabras, nunca es necesario utilizar una sentencia GOTO dentro de un programa, y esto, en programación estructurada, se ha dado en llamar incorrectamente «el arte de no utilizar el GOTO». En la práctica, el GOTO se considera más como algo peligroso que como algo prohibido.

En los casos en que una sección de un programa es clara en el sentido de saber cuándo se ejecuta cada sentencia, todo resulta correcto, y puede ser que la mejor forma de asegurar una mayor claridad sea utilizando una sentencia GOTO.

Otro elemento importante para conseguir la claridad de los programas, es utilizar nombres de variables y subrutinas que signifiquen «algo» de verdad. Por ejemplo, si una subrutina imprime un tablero de ajedrez sobre el que jugar, ésta deberá llamarse algo así como: 100 REM pintar tablero, en lugar de 100 REM xy4. Si llega a ser capaz de pensar nombres apropiados para variables y subrutinas, entonces, los programas que escriba serán más fáciles de entender. En la práctica, generalmente es muy difícil inventar el número necesario de nombres, y conseguir que además sean cortos, ya que no hay nada peor que tener que teclear una y otra vez en un programa un nombre de variable excesivamente largo. Se puede llegar a ser bastante hábil dando nombres a va-

riables y subrutinas, o bastante inhábil (como la mayoría de los programadores), pero, definitivamente, el esfuerzo merece la pena.

Todo lo dicho sobre claridad y estilo puede resultar aburrido, especialmente si siempre se ha pensado que el aspecto más importante de la programación es la eficacia; esto es, conseguir la máxima velocidad de ejecución empleando la mínima memoria. En general, en cualquier programa práctico, la eficacia es lo más importante, y el estilo suele pasar a un segundo lugar. Pero, en cualquier caso, si uno llega a ser capaz de producir un programa rápido, con memoria libre para adiciones y modificaciones, pero que nadie es capaz de entender (ni siquiera uno mismo pasados unos días), entonces las adiciones y modificaciones nunca se harán , y muchos errores podrán quedarse en el programa.

El énfasis se debe poner siempre en la claridad, y cualquier arreglo para aumentar la velocidad de ejecución y emplear menos memoria, no será más que una modificación sobre un programa bien hecho.

Por motivos de claridad, el uso de código máquina se ha reducido al mínimo. Los únicos programas que incluyen código máquina son «Haciendo la rana» y «Serpientes y Escalerillas».

Convenios relativos a los listados

Todos los programas que se incluyen, han sido comprobados exhaustivamente tal y como aparecen. Deliberadamente se ha decidido no renumerarlos una vez que están en la forma final, que funciona perfectamente. Esto se ha hecho para evitar errores. Los listados de los programas provienen de la impresora del ZX, lo cual implica que podrían aparecer en una forma más fácil de leer, con divisiones en los puntos en que se haya de conservar la sintaxis y líneas blancas insertadas entre las subrutinas. Se han producido listados que señalan cómo se tienen que teclear los programas, más que indicar cómo han de aparecer una vez tecleados. Por ejemplo, las palabras clave GOSUB o GOTO aparecen como una sola palabra, tal como están en el teclado del Spectrum, en lugar de aparecer como dos palabras, que es el modo en que aparecen en la pantalla después de pulsar ENTER. Al teclear los programas en el Spectrum, no es necesario dejar espacios después de las palabras clave, porque el ordenador lo hace por sí mismo. Por supuesto, en el único caso en que se necesita pulsar SPACE es cuando queremos obtener un espacio en blanco en una sentencia PRINT.

Todos los caracteres que deben ser introducidos en modo gráfico, han sido encerrados entre corchetes. Es decir, si hemos de teclear la letra «A» en modo gráfico, aparecerá como [A] en el listado.

Cuando sea necesario mantener pulsada la tecla CAPS-SHIFT en el modo gráfico para obtener determinados carácteres (los que están en las

teclas de los números), el símbolo ^ aparecerá inmediatamente delante del número en cuestión. Por ejemplo, [^ 1] indica que hemos de pulsar la tecla del 1 en modo gráfico, manteniendo pulsada la tecla CAPS-SHIFT.

Técnicas de los juegos

Finalmente, antes de comenzar con el primer programa, daremos un vistazo a cada uno de los juegos, y a las técnicas que requieren.

El juego del capítulo 2, «La colina de las hormigas», es un juego simple de animación en una sola dimensión y sirve para introducir los «sprites» y el bucle de animación. El capítulo 3 presenta un juego llamado «El salto de la rana», introduce el movimiento en dos dimensiones y controla el modo en que un objeto bota alrededor de la pantalla sometido a la acción de la gravedad.

El capítulo 4 utiliza un tipo diferente de animación, la animación con deslizamiento de pantalla en el juego «Haciendo la rana», que es una versión de otro juego «de marcianitos» muy conocido.

Los capítulos 5 y 6 tratan de uno de los objetos más largos e interesantes que se pueden animar muy fácilmente, «La serpiente». El capítulo 5 desarrolla el juego de la serpiente básico, considerando los problemas de mover eficazmente una serpiente, e introduce la idea de una «cola». El capítulo 6 extiende este juego incluyendo varios sprites adicionales, y el resultado es «Los renacuajos».

El capítulo 7 se sirve de un juego tradicional, «Serpientes y Escalerillas», para escribir una versión actualizada del mismo. La técnica más importante explicada como parte de este último juego, es la animación de un «sprite» sobre un fondo bastante complicado, que no se destruye cada vez que el «sprite» se mueve.

Cada uno de los juegos introduce alguna idea nueva relativa a las técnicas de animación y al uso del ordenador, que se unifican en el capítulo final con comentarios acerca del diseño y escritura de los juegos en general, utilizando el Spectrum en particular. Cuando se ha llegado a este punto, se consigue un gran avance en ambos campos y se puede disfrutar mucho más de los juegos.

2. La colina de las hormigas

La colina de las hormigas es un juego muy simple, pero efectivo, de animación de varios objetos.

La idea básica es guiar a un hombre a través de los túneles que pertenecen a una colonia de hormigas, con el fin de alcanzar y destruir el nido de huevos, que está en el punto más profundo. La dificultad aumenta, porque el tiempo es limitado, y además hay que esquivar a las hormigas soldado colocadas en cada nivel de las galerías. El mayor problema para escribir un juego de esta clase está en conseguir la animación de varios objetos, —el hombre y las hormigas—, al mismo tiempo. En este capítulo trataremos este problema, y también se desarrollarán métodos estándar que utilizaremos posteriormente.

Diseño del juego

Antes de empezar a escribir un programa, es una buena idea especificar con el mayor detalle posible qué es lo que queremos que haga. Los programas de juego son ligeramente diferentes de otras aplicaciones, ya que en general no es posible describir con exactitud el juego final, a menos que se haya escrito algo del mismo. Esto es debido a la dificultad de predecir cómo funcionan los distintos elementos, si no se han probado todavía. Incluso después de haber escrito muchos programas, sigue siendo difícil anticipar el efecto final a partir de la combinación de los diferentes elementos (tomados de juegos ya existentes), para escribir uno nuevo.

De todos modos, es bastante útil anotar qué es lo que se espera de un juego antes de empezar a escribirlo, ya que así tendremos la ocasión de mejorarlo.

El diseño de la Colina de las Hormigas (y de otros muchos juegos dinámicos) tiene 3 partes:

1. Los gráficos del fondo
2. Los caracteres móviles y las reglas de movimiento
3. Las consecuencias de ganar y perder.

El telón de fondo consiste en varios túneles horizontales conectados por galerías verticales (Figura 2.1). La razón de utilizar una o dos galerías para conectar los túneles es dar al juego mayor variedad de estrategias.

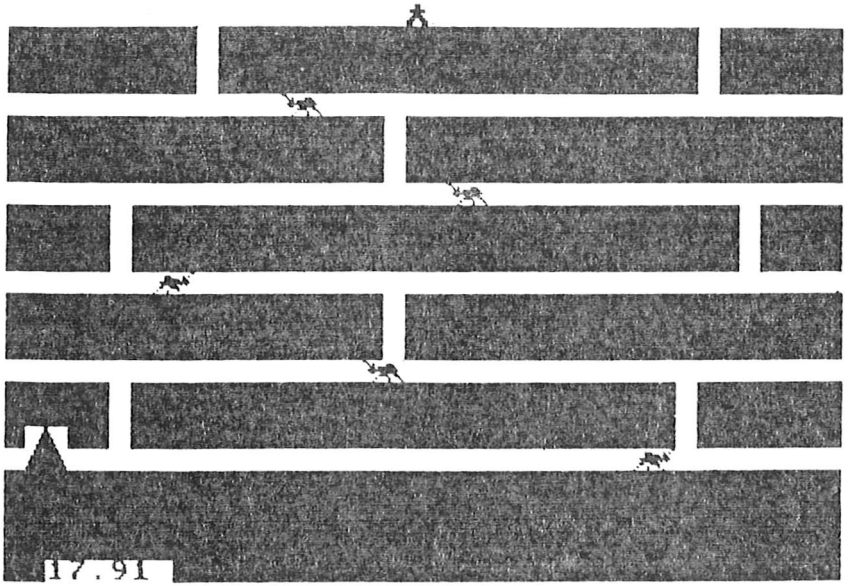


Fig. 2.1

Cuando sólo hay una galería que conecta dos niveles, únicamente hay una ruta posible, y el juego se convierte en un cronometraje, pero si hay dos galerías, el jugador tiene la oportunidad de elegir por cuál de ellas bajar. La forma del fondo nos sugiere que al menos necesitaremos 3 colores: uno para los túneles, seguramente negro, otro para la tierra, probablemente rojo, y un tercero para el cielo, casi seguro el azul. También necesitamos asignar colores al hombre y a las hormigas. La figura del hombre se puede almacenar como un único carácter de gráfico definido por el usuario (Fig. 2.2). Como sólo se utiliza un carácter, los túneles y las galerías no tendrán más que un carácter de anchura.

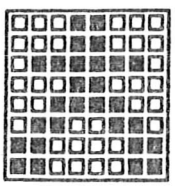
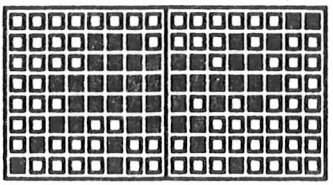


Fig. 2.2 *Carácter gráfico para el hombre.*

Las hormigas son mucho más difíciles de almacenar en un solo carácter de 8×8 puntos, porque son más largas y delgadas. La solución está en utilizar dos gráficos de usuario. (Fig. 2.3). Puesto que las hormigas sólo se mueven horizontalmente a lo largo de los túneles, y nunca bajan

por las galerías, aunque estén formadas de dos caracteres, basta con que las galerías y túneles sólo tengan un carácter. Las reglas de movimiento para las hormigas son fáciles de definir.



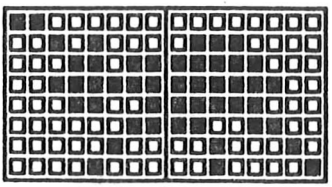


Fig. 2.3 *Caracteres para las hormigas.*

Cada túnel debe tener una hormiga soldado confinada en ese nivel, y sólo puede moverse a lo largo del túnel. Si una hormiga llega a alcanzar al hombre, entonces éste es capturado y el juego se acaba. En este punto es un poco difícil decir exactamente cómo deberían moverse las hormigas para que el juego fuese bueno, pero parece razonable empezar asumiendo que éstas se mueven aleatoriamente a lo largo del túnel, y que ocasionalmente cambian de dirección, también al azar. Esta elección de «hormigas aleatorias» dará al juego la dificultad suficiente, siempre que el tiempo límite para acabarse sea adecuado.

Las reglas de movimiento del hombre son un poco más complicadas. Del mismo mdo que las hormigas, puede moverse a lo largo de los túneles, pero también puede bajar por las galerías. El juego sería probablemente demasiado fácil si el hombre pudiese esperar en las galerías, por lo tanto, éste bajará directamente de un nivel a otro. Probablemente también sería demasiado fácil si el hombre pudiese subir por las galerías, por tanto, el control del jugador sobre el hombre está limitado a las teclas flecha derecha e izquierda para el movimiento horizontal, y a la tecla de la flecha hacia abajo para bajar por las galerías. Si se pulsa la tecla de la flecha de bajada cuando el hombre está en la boca de una galería, entonces podrá «saltar» al siguiente nivel. Pero si no es así, el pulsar dicha tecla no servirá para nada.

Aparte de los detalles del estilo de no permitir que el hombre o las hormigas se muevan fuera del borde de la pantalla, esto completa la descripción del movimiento.

Hay tres formas posibles de terminar el juego:

1. El hombre alcanza el nido
2. Una hormiga captura al hombre
3. El tiempo límite es sobrepasado y los huevos estallan.

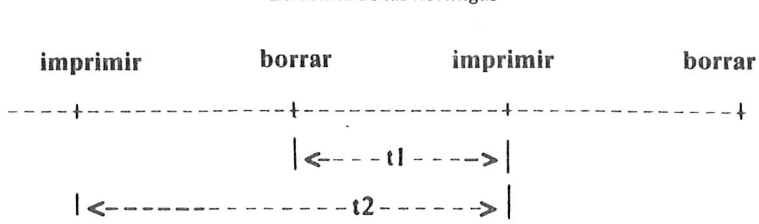
Aunque no hay duda de que para que un juego tenga éxito éste ha de ser muy divertido cuando se está jugando, una parte importante puede ser el construir finales interesantes. Por ejemplo, en este caso, cuando el hombre alcanza el nido, éste podría ser recompensado con sonido de trompetas y el destello de la pantalla, cuando una hormiga captura al hombre, ésta podría arrastrarlo hasta el nido, y cuando se alcanza el tiempo límite, se podría señalar con una explosión de la población de hormigas.

Una vez decidido el diseño del juego, es tiempo ya de pasar a escribir el programa, pero primero es conveniente volver sobre algunas ideas y dificultades de la animación por ordenador.

La técnica de la animación

La animación de figuras pequeñas en un ordenador es muy simple. Para conseguir que una figura cualquiera se mueva, basta con repetir lo siguiente: se imprime en un determinaddo lugar, después se borra (generalmente imprimiendo un blanco), y luego se vuelve a imprimir en su nueva posición. Si se puede repetir el ciclo «imprimir-borrar-imprimir» con la suficiente rapidez, se crea la ilusión de un suave movimiento. En la práctica hay 2 factores que controlan la suavidad del movimiento generado por la computadora. El primero es el lapso de tiempo entre borrar y reimprimir la figura, que llamaremos t1, y que debe ser siempre lo más corto posible, porque es el intervalo de tiempo en que la figura no es visible en la pantalla. Si es demasiado largo, el objeto parece centellear. El segundo factor es el intervalo de tiempo entre el principio de cada impresión de la figura, que llamaremos t2 y puede ser tan largo como se quiera, porque controla la rapidez del movimiento. En la mayor parte de los casos, se quiere que el objeto se mueva rápidamente, y, por tanto, t2 deberá ser también corto, pero no es siempre así. La forma en que los diferentes tiempos afectan a la animación se puede ver en la figura 2.4.

Hasta ahora hemos ignorado otro factor que también controla la velocidad del movimiento. Hemos asumido que cada vez que el objeto se mueve por la pantalla lo hace el más pequeño espacio posible, es decir un solo carácter. La ventaja de hacerlo así es que el movimiento parece más suave y el objeto se imprime en todas las posiciones que atraviesa. Por otro lado es una gran ventaja mover el objeto más de una posición de carácter cada vez, y esta ventaja es la velocidad. El BASIC es suficiente para conseguir que un objeto parezca moverse bastante deprisa,



t1 debe ser siempre lo más pequeño posible.
t2 controla la velocidad del movimiento.

Figura 2.4.

sin más que mover más de una posición de carácter cada vez. En «La Colina de las Hormigas» se utilizan ambos métodos de animación. El hombre se moverá de uno en uno, pero a veces las hormigas se mueven distancias mayores de un golpe.

El ciclo «imprimir—borrar—imprimir» es la base de los gráficos animados, pero todavía nos queda el problema de conseguir mantener el movimiento dentro del programa. Cada objeto de la pantalla está asociado con 5 cantidades, y, por tanto, 5 variables: El carácter que define su forma; sus coordenadas de posición X e Y, y las cantidades que X e Y deben cambiar cada vez dentro del bucle de animación.

Por ejemplo, la forma de un objeto puede ser definida por un carácter almacenado en C$, su posición en X e Y, y su velocidad en VX, VY. El bucle de animación puede ser:

```
PRINT AT Y, X;" ";: REM borrar objeto
LET X=X+VX: LET Y=Y+VY: REM actualizar coordenadas
PRINT AT Y, X; C$;: REM reimprimir objeto
```

En la práctica, los bucles de animación son generalmente un poco más complicados, porque la velocidad de los objetos cambia como consecuencia de su situación en la pantalla. Por ejemplo, la animación de una pelota que bota supone un cambio de velocidad cada vez que ésta choca con el borde de la pantalla.

Esta es toda la teoría de la animación simple. Un objeto que se anima de este modo se suele denominar «sprite». El programa utiliza un total de 6 «sprites», 5 hormigas y un hombre, y es una buena demostración del hecho de que, en computación, la teoría no es más que una serie de principios generales.

El programa principal

El programa principal de todos los juegos dinámicos es casi siempre muy parecido. Usualmente hay una inicialización, seguida del bucle de animación, y después las rutinas que conectan con el final del juego. La Colina de las Hormigas no es una excepción:

```
 1Ø REM el monte de la hormiga
 2Ø CLS
 3Ø GO SUB 1ØØØ
 4Ø GO SUB 8ØØØ
 5Ø GO SUB 2ØØØ
 6Ø REM bucle
 7Ø GO SUB 4ØØØ
 8Ø IF TIMEUP OR HIT OR HOME
     THEN  GO TO 1ØØ
 9Ø GO TO 6Ø
1ØØ GO SUB 6ØØØ
11Ø CLS
199 STOP
```

Las líneas 30 a 50 son subrutinas que establecen el comienzo del juego, de la 60 a 90 forman el bucle de animación, y las líneas 100 a 199 nos llevan al final del juego. La subrutina 1000 es la encargada de hacer cosas tales como establecer los gráficos de usuario, e inicializar todas las variables que han de ser inicializadas. La subrutina 8000 simplemente imprime un título y fija el nivel de dificultad del juego. La subrutina 2000 se ocupa de dibujar el conjunto de túneles y galerías, de modo que el juego queda listo para empezar. Se podría decir que el corazón del juego es la subrutina 4000, ya que es la responsable de mover al hombre y a las hormigas. Esta rutina se ejecuta repetidamente, hasta que una de las 3 variables TIMEUP, HIT o HOME toma el valor VERDADERO, que en el Spectrum significa NO—CERO. Estas 3 variables se utilizan para indicar las 3 posibles razones de que el juego acabe, y esto se tiene en cuenta dentro de la subrutina 4000. Hay que resaltar que cualquier variable númerica puede almacenar los valores lógicos VERDADERO o FALSO , y se puede utilizar en subrutinas en las que sea necesario un test de comprobación. Ahora lo único que nos queda por hacer es escribir cada una de las subrutinas del programa principal.

Subrutinas 1000 y 8000

Las 2 primeras subrutinas en ser llamadas, y que empiezan en las líneas 1000 y 8000, no han sido difíciles de escribir. Sin embargo, la

subrutina 1000 ha sido una de las pocas que ha tenido que ser cambiada durante el desarrollo del programa, se le han añadido algunas líneas, y también se ha tenido que volver a numerar. La razón de esto fue que no estaba del todo claro qué variables se debían haber inicializado hasta que otras subrutinas fueron escritas. Por tanto, resulta más claro dar la versión final de la subrutina, que presentar la primera versión de la subrutina 1000 y después introducir los cambios según vayamos escribiendo el resto de las subrutinas.

```
1000 REM inicializacion
1010 FOR Q=97 TO 105
1020 FOR R=0 TO 7
1030 READ D
1040 POKE USR (CHR$ (Q))+R,D
1050 NEXT R
1060 NEXT Q
1070 REM datos
1080 DATA 0,14,15,63,31,51,68,
          132
1090 DATA 3,20,216,252,168,192,
          64,16
1100 DATA 0,112,248,252,248,204,
          34,33
1110 DATA 192,40,27,63,25,3,4,8
1120 DATA 24,24,126,24,60,102,
          195,195
1130 DATA 1,1,3,3,7,7,15,15
1140 DATA 31,31,63,63,127,127,
          255,255
1150 DATA 128,128,192,192,224,
          224,240,240
1160 DATA 248,248,252,252,254,
          254,255,255
1170 LET XM=INT (RND*20)+10
1180 LET YM=1
1190 DIM A(5,3): DIM A$(5,2)
1200 FOR Q=1 TO 5
1210 LET A(Q,1)=INT (RND*25)+5
1220 LET A(Q,2)=Q*4+1
1230 LET A(Q,3)=1
1240 LET A$(Q)="[AB]"
1245 IF RND*1>0.5
      THEN  GO SUB 3500
1250 NEXT Q
1260 LET HIT=0
1270 LET HOME=0
```

```
1280 LET TIMEUP=0
1299 RETURN
```

La primera parte de la subrutina 1000 crea los gráficos de usuario necesarios. Las líneas 1010 a 1060 ejecutan el bucle que lee los datos en las líneas 1080 a 1160 y utiliza estos datos para crear los caracteres. Las líneas 1080 y 1090 contienen los datos para crear una hormiga mirando hacia la derecha, y las líneas 1100 Y 1110 guardan los datos para una hormiga mirando hacia la izquierda. Esto es, siguiendo la subrutina 1000:

```
PRINT AT Y,X;"[AB]";
```

Imprimirá una hormiga en X, Y mirando a la derecha y

```
PRINT AT Y,X;"[DC]";
```

imprimirá una hormiga mirando a la izquierda en el lugar X, Y; la necesidad de utilizar 2 tipos de hormigas no ha sido mencionada hasta ahora, pero es debido a que las hormigas se mueven hacia la derecha y a laizquierda. Si sólo se utilizase un tipo para ambas direcciones, a veces la hormiga andaría hacia atrás. Se podría pensar que el señalar qué dos caracteres de hormigas hay que utilizar en cada caso, es algo difícil, sin embargo, mirando la subrutina 3000, comprobaremos que es bastante fácil.

La línea 1120 guarda los datos para el carácter del hombre como gráfico en E, y las líneas 1130 a 1160 almacenan los datos de los Y caracteres que unidos nos dan la pirámide que va a representar el nido. Es decir:

```
PRINT AT Y-1,X;"[FG]";
PRINT AT Y,X;"[HI]";
```

Imprimirá los 4 caracteres que forman el nido, con su esquina inferior izquierda en X, Y. El resto de la subrutina 1000 crea e inicializa las variables que más tarde utilizaremos en el programa. Las líneas 1170 y 1180 establecen la posición inicial del hombre en XM e YM . La posición vertical del hombre siempre empieza en 1, y, por tanto, está sobre el suelo, pero la posición horizontal es aleatoria. Las líneas 1200 a 1250 inicializan las posiciones de las 5 hormigas soldado, su dirección de movimiento, y su forma (es decir, mirando a la izquierda o a la derecha). La matriz A(Q,2) nos da la coordenada y asociada, y A(Q.3) nos da la dirección de movimiento de la hormiga. Si A(Q,3) es 1, entonces la hormiga Q se mueve hacia la derecha, y si es -1, entonces la hormiga Q se mueve hacia la izquierda. Finalmente, el par correcto de caracteres que forman la hormiga se almacena en A$(Q). Por tanto, para imprimir la hormiga Q haremos lo siguiente:

```
PRINT AT A(Q,2),A(Q,1);A$(Q);
```

La línea 1210 inicializa la posición vertical de cada una de las hormigas, ya que hay una hormiga por túnel. Esto es, la hormiga 1 se imprime

en la línea 5, la 2 en la 9, la 3 en la línea 13....y así todas. Todas las direcciones iniciales se establecen como 1, haciendo que todas las hormigas se muevan hacia la derecha (línea 1230) y, por tanto, la línea 1240 almacena una figura de hormiga mirando a la derecha en A$(Q) para todas las hormigas que se muevan hacia la derecha. La parte final de la subrutina 1000 pone FALSO (es decir Ø) en las variables que se utilizarán para indicar el final del juego (líneas 1260 a 1280).

La subrutina título es muy sencilla. Por supuesto, en ella se deben escribir las instrucciones claras para explicar el juego a los jugadores nuevos.

```
8ØØØ REM titulo
8Ø1Ø INK 2: PAPER 6: CLS
8Ø2Ø PRINT AT 3,6;
     "MONTE DE LA HORMIGA"
8Ø3Ø FOR I=1 TO 16: PRINT A$(1):
      NEXT I
8Ø4Ø PRINT AT 5,2;
     "En este juego tienes un ";
     "tiempo"
8Ø5Ø PRINT AT 6,2;"fijo para ";
     "destruir el nido de "
8Ø6Ø PRINT AT 7,2;"hormigas ";
     "antes de que las"
8Ø7Ø PRINT AT 8,2;"crias ";
     "salgan del cascaron!"
8Ø8Ø PRINT AT 9,2;"Usa los ";
     "cursores para moverte";
     AT 1Ø,2;"pero"
8Ø9Ø PRINT AT 11,2;"CUIDADO:";
     "las hormigas soldado"
81ØØ PRINT AT 12,2;"que ";
     "guardan los tuneles te "
81Ø5 PRINT AT 13,2;"apresaran";
     " y te llevaran al"
811Ø PRINT AT 14,2;"nido para ";
     "alimentar a sus ";
     AT 15,2;"crias"
812Ø PRINT AT 16,2;"Buena ";
     "suerte"
82ØØ PRINT AT 18,2;"Nivel de ";
     "dificultad_"
821Ø PRINT AT 19,2;"1.Experto,";
     " 2.Medio, 3.novato ?";
822Ø INPUT DF
```

```
8230 IF DF<1 OR DF>3
      THEN  GO TO 8200
8240 LET MAX=(5+DF*20)+
     INT (RND*10)
8299 RETURN
```

Las líneas 8200 a 8240 piden que el usuario elija el nivel de dificultad del juego. La línea 8240 comprueba que DC está dentro del intervalo de posibilidades correcto, y la línea 8340 establece en MAX el tiempo del que dispone el jugador de acuerdo con el nivel de dificultad elegido.

Subrutina 2000

La subrutina 2000 es la primera que, por fin, realiza algo relacionado con los gráficos. Su trabajo es dibujar el sistema de túneles y galerías, y prepararnos a empezar el juego. Puesto que sólo se utiliza una vez en todo el programa, su velocidad no es algo importante, y lo que nos interesa es su claridad y que sea una subrutina fácil de modificar.

```
2000 REM dibujo del hormiguero
2010 PAPER 3
2020 BORDER 3
2030 CLS
2040 PAPER 0
2050 FOR X=0 TO 31
2060 PRINT AT 5,X;" "
2070 PRINT AT 9,X;" "
2080 PRINT AT 13,X;" "
2090 PRINT AT 17,X;" "
2100 PRINT AT 21,X;" "
2110 NEXT X
2120 PAPER 5
2130 FOR X=0 TO 31
2140 FOR Y=0 TO 1
2150 PRINT AT Y,X;" ";
2160 NEXT Y
2170 NEXT X
2180 PAPER 0
2190 FOR I=1 TO 5
2200 READ NUM,Y1,Y2
2230 GO SUB 9000
2240 NEXT I
2250 DATA 2,2,4
2260 DATA 1,6,8
```

```
2270 DATA 2,10,12
2280 DATA 1,14,16
2290 DATA 2,18,20
2300 INK 3: PAPER 5
2310 PRINT AT YM,XM;"[E]";
2320 INK 0: PAPER 3
2330 PRINT AT 20,1;"[FH]"
2340 PRINT AT 21,1;"[GI]"
2350 PAPER 0
2360 FOR I=1 TO 5
2370 PRINT AT A(I,2),A(I,1);
      INK 6;A$(I);
2380 NEXT I
2390 POKE 23672,0: POKE 23673,0:
      POKE 23674,0
2399 RETURN
```

Las líneas 2010 a 2030 tienen la pantalla de magenta, y las líneas 2040 a 2110 imprimen los túneles horizontales en negro, imprimiendo 5 líneas horizontales de espacios. Esto produce 5 bandas negras sobre el suelo magenta. Después, las líneas 2120 a 2170 imprimen un bloque de espacios cyan para representar el cielo. Unicamente nos queda imprimir las galerías verticales que conectan los túneles. Este problema se resuelve mejor llamando a otra subrutina, utilizando los datos de las líneas 2250 a 2290. La subrutina 9000 dibujará NUM galerías verticales, de modo que cada una de ellas empieza en Y1 y termina en Y2.Así, por ejemplo, utilizando los datos 10,12, la subrutina dibuja dos galerías entre los túneles en Y = 10 e Y = 12.

Las líneas 2300 y 2310 imprimen la figura del hombre en YM, XM utilizando tinta magenta y papel cyan, ya que, de momento, el hombre está sobre el suelo. Más tarde, cuando esté en los túneles o galerías, el color del fondo será negro. Las líneas 2320 a 2340 imprimen el nido en la esquina inferior izquierda del último túnel. Por fin, las líneas 2360 a 2380 imprimen las hormigas, y la línea 2390 pone a cero el tiempo para que el juego empiece.

La única subrutina empleada por la subrutina 2000 es la subrutina 9000, que hemos de completar antes de pasar a otras subrutinas:

```
9000 REM pozos
9010 LET X1=INT (RND*10)+4
9020 LET X2=INT (RND*10)+16
9030 FOR Y=Y1 TO Y2
9040 IF NUM=1 THEN
```

```
      PRINT AT Y,((X1+X2)/2);
      " ";
9050 IF NUM=2 THEN
      PRINT AT Y,X1;" ";AT Y,X2;
      " ";
9060 NEXT Y
9099 RETURN
```

La línea 9010 fija aleatoriamente las posiciones de las dos galerías en X1 yX2. Los números aleatorios se generan de modo que la galería X1 está a la izquierda en la pantalla, y la X2 a la derecha. Si necesitamos dos galerías, ambas son impresas por la línea 9050 con el bucle FOR (líneas 9030 a 9060). Si sólo se necesita una, se imprimirá en el punto medio de X1 y X2, que es una posición aleatoria, pero cercana al centro de la pantalla (línea 9040).

Subrutina 4000 —El corazón del juego

La subrutina más importante de todas es la 4000, Como va a ser llamada tantas veces, es importante que tanto ella como todas las subrutinas a las que llame sean lo más rápidas posible. Una vez dicho esto, sigue siendo necesario intentar escribir las cosas sencilla y claramente, y con este fin escribiremos la subrutina 4000, con una serie de llamadas a otras subrutinas más cortas:

```
4000 REM un movimiento
4010 GO SUB 5000
4015 IF HIT=1 THEN  GO TO 4099
4020 GO SUB 3000
4030 GO SUB 4500
4099 RETURN
```

La tarea de la subrutina 5000 es dar al jugador la oportunidad de mover al hombre, y comprobar el resultado de este movimiento en el acercamiento al nido. Del mismo modo, la subrutina 3000 está relacionada con el movimiento de las hormigas, y la comprobación de si el hombre ha sido capturado. La subrutina 4500 imprime el tiempo transcurrido desde el comienzo del juego, y comprueba si éste se ha agotado. Es claro que esta forma de escribir la subrutina 4000 pospone todas las decisiones difíciles hasta el momento de escribir las otras subrutinas, pero esto es intencionado, los problemas deben ser siempre resueltos paso a paso.

La siguiente subrutina que escribimos es la 5000:

```
5000 REM mover el hombre
5010 PAPER 0
```

```
5020 IF YM<2 THEN  PAPER 5
5030 PRINT AT YM,XM;" ";
5040 IF INKEY$="5" AND XM>0
      THEN  LET XM=XM-1
5050 IF INKEY$="8" AND XM<31
      THEN  LET XM=XM+1
5060 IF INKEY$="6" AND
       ATTR (YM+1,XM)=2
      THEN  GO SUB 5500
5070 PRINT AT YM,XM; INK 3;
     "[E]";
5080 IF XM=3 AND YM=21
      THEN  LET HOME=1
5099 RETURN
```

La línea 5020 se ocupa de ver cuál es la posición vertical del hombre. Si éste está todavía sobre el suelo, el color del fondo se fija en cyan, y, si está bajo tierra, el color del fondo se fija en negro. La línea 5030 borra el hombre de su actual posición YM, XM. Las líneas 5040 a 5060 controlan cada una de las 3 posibles teclas de flecha izquierda, la 5050 la derecha, y la 5060 controla la tecla de bajada. También las líneas 5040 y 5050 nos aseguran que el hombre no se sale fuera de la pantalla. La línea 5060 llama a la subrutina 5500 para que el hombre salte por una galería solamente si la tecla de bajada se pulsa cuando el hombre está situado sobre un carácter negro. La función ATTR se utiliza para descubrir el color del carácter que está justamente debajo del hombre.

Sólo falta la subrutina 5500 para que la 5000 funcione:

```
5500 REM bajar por el pozo
5510 PAPER 0: INK 3
5520 FOR I=1 TO 4
5530 LET YM=YM+1
5540 PRINT AT YM,XM;"[E]";
5550 BEEP .075,0-(I*4+10)
5560 PRINT AT YM,XM;" ";
5570 NEXT I
5599 RETURN
```

Esta empieza estableciendo el color del fondo como negro (línea 5510), porque el color de las galerías siempre es negro. Después, las líneas 5520 a 5570 bajan el hombre por la galería carácter a carácter.

La subrutina 3000 se enfrenta con el primer problema real; tenemos un solo hombre, pero hay 5 hormigas. Si la subrutina moviese las 5 hormigas cada vez que es llamada, entonces la figura del hombre respondería muy lentamente a las órdenes del usuario, porque mover cinco hormigas requiere su tiempo. La solución obvia es mover única-

mente una hormiga que seleccionamos al azar, para cada movimiento del hombre. Esto es exactamente lo que hace la subrutina.

```
3000 REM  mover la hormiga
3010 PAPER 0
3020 LET Q=INT (RND*5)+1
3030 PRINT AT A(Q,2),A(Q,1);
     "  ";
3040 IF RND*1<.01
      THEN  GO SUB 3500

3050 LET A(Q,1)=A(Q,1)+A(Q,3)*RND*3
3060 IF A(Q,1)<3 THEN LET A(Q,1)=3:GOSUB 3500
3070 IF A(Q,1)>27 THEN LET A(Q,1)=27:GOSUB 3500

3080 PRINT AT A(Q,2),A(Q,1);
      INK 6;A$(Q)
3085 INK 7
3090 GO SUB 9200
3099 RETURN
```

La línea 3020 fija en Q un número aleatorio entre 1 y 5, y determina qué hormiga va a moverse. Entonces la línea 3030 borra la hormiga de su posición. La línea 3040 llama al azar a la subrutina 3500 para invertir la dirección de la hormiga Q. RND*1 es más pequeño que 0,01, una de cada 100 llamadas a la subrutina 3000. La línea 3050 aumenta o disminuye la coordenada X de la hormiga, dependiendo del valor almacenado en A(Q,3). Si A (Q,3) es +1, entonces RND*3 se suma a la coordenada X, y la hormiga se mueve hacia la derecha. Si A(Q,3) es -1, entonces RND*3 se resta de la coordenada X, y la hormiga se mueve hacia la izquierda.

El espacio que recorre la hormiga, viene dado por el valor RND*3. Esto quiere decir que cada vez avanzará 1, 2 ó 3 posiciones de carácter.

Aunque el mover los objetos más de una posición de carácter cada vez suele ser una fuente de problemas, es la única forma de dar la impresión de que las hormigas se mueven deprisa.

Después de actualizar la coordenada X de la hormiga, ésta es registrada por las líneas 3060 y 3070 para comprobar si la hormiga se va a

salir de la pantalla. Si es así, la subrutina 3500 es llamada para cambiar la dirección de la hormiga. Finalmente, la línea 3080 reimprime la hormiga en la nueva posición, y la línea 3090 utiliza la subrutina 9200 para controlar si la hormiga que acaba de moverse ha capturado al hombre.

Todo lo que la subrutina 3500 tiene que hacer es cambiar la dirección de movimiento de la hormiga, y el par de caracteres gráficos.

```
3500 REM cambio de sentido
3510 IF A$(Q)="[AB]"
      THEN  LET A$(Q)="[DC]":
        GO TO 3530
3520 LET A$(Q)="[AB]"
3530 LET A(Q,3)=-A(Q,3)
3599 RETURN
```

La subrutina 9200 es tambien fácil de escribir:

```
9200 REM atraparon al hombre
9210 IF A(Q,2)<>YM THEN  RETURN
9220 IF A(Q,1)<>XM AND
     A(Q,1)+1<>XM THEN  RETURN
9230 LET HIT=1
9299 RETURN
```

La única subrutina de las que usa la 4000 que queda por ver, es la 4500.

```
4500 REM tiempo
4510 LET T=(PEEK 23672+
           256*PEEK 23673+
           65536*PEEK 23674)/100
4520 PRINT AT 0,27;INT (T);
4530 IF T>MAX THEN  LET TIMEUP=1
4599 RETURN
```

La línea 4520 imprime el tiempo actual, calculando en la línea 4510, y la línea 4530 comprueba si el tiempo se ha agotado y un juego se ha terminado.

Ahora que la subrutina 4000 y todas las que utiliza se han escrito, el juego puede ser probado, exceptuando las subrutinas de final de juego.

Subrutina 6000 - Sección de fin del juego

La elección del final de un juego suele estar limitada por la cantidad de tiempo que ya han empleado las partes principales del programa al ser ejecutadas. Hay tres posibilidades de concluir este juego, y la subrutina 6000 tiene que indentificarlas y llamar a las subrutinas adecuadas:

```
6000 REM final del juego
6010 IF HIT THEN  GO SUB 6200:
       GO TO 6100
6020 IF TIMEUP THEN  GO SUB 6500
     : GO TO 6100
6030 IF HOME THEN  GO SUB 6800
6100 FOR I=1 TO 100: NEXT I
6105 PAPER 7
6110 INK 0
6120 BORDER 7
6130 CLS
6140 INPUT "Otra partida s/n?";
     N$
6150 IF N$="s" THEN  RUN
6160 IF n$<>"n" THEN  GO TO 6140
6199 RETURN
```

La subrutina 6200 es llamada si una hormiga ha llegado a una posición que ocupa el hombre en la pantalla, la 6500 es llamada si el juego termina porque se ha agotado el tiempo límite, y la 6800 es llamada si el hombre alcanza el nido dentro del tiempo permitido. Después de la llamada a una de estas subrutinas, las líneas 6140 a 6160 preguntan al jugador si desea volver a jugar.

La más complicada de las subrutinas llamadas por la subrutina 6000 es la 6200, que luego será descrita. La subrutina 6500 simplemente imprime hormigas en posiciones aleatorias acompañadas de FLASH y de ruidos.

```
6500 REM salir del cascaron
6510 PRINT AT 21,1;
     "LAS HORMIGAS SALEN DEL ";
     "CASCARON"
6520 FOR I=1 TO RND*20+20
6530 LET X=RND*25+5
6540 LET Y=RND*17+3
6550 PRINT AT Y,X; FLASH 1;
     A$(INT (RND*5)+1);
6560 BEEP .1,RND*20-30
6570 NEXT I
6599 RETURN
```

La subrutina 6800 recompensa al jugador con un simple sonido de trompetas:

```
6800 REM destruir nido
6805 RESTORE 6820
6810 PRINT AT 21,1;
     "LO CONSEGUISTE"
```

```
6820 DATA 5,.25,7,.25,8,.25,11,
     .25,12,.5,12,.5
6830 DATA 999,999
6840 READ P,D
6850 IF P=999 THEN  GO TO 6899
6860 BEEP D,P
6870 GO TO 6840
6899 RETURN
```

Las notas de este sonido están especificadas por los valores de las sentencias DATA (líneas 6820 y 6830). Cada nota corresponde a un par de valores, el primero es el tono, y el segundo la duración. El tono 999 se emplea para marcar el final de la secuencia de notas, y es detectado por la línea 6850.

La idea escrita en la subrutina 6200 es que la hormiga que captura al hombre lo lleva hasta la galería más próxima, y lo deja caer para que sea recogido por la hormiga del siguiente nivel. Esto se repite hasta que la hormiga del último nivel lleva al hombre hasta el nido, y el juego acaba.

Esto puede parecer un poco complicado, pero, en la práctica, no lo es:

```
6200 REM captura
6210 PRINT AT 0,10;
     "HAS SIDO CAPTURADO"
6220 IF Q=5 THEN  GO TO 6320
6230 GO SUB 5900
6240 GO SUB 5600
6250 LET X=X1
6260 IF ABS (X2-A(Q,1))<
        ABS (X1-A(Q,1))
      THEN  LET X=X2
6270 GO SUB 5800
6280 PRINT AT A(Q,2),A(Q,1)+2;
     " ";
6290 BEEP .1,-10
6295 GO SUB 5500
6296 PRINT AT YM,XM;"[E]";
6300 LET Q=Q+1:
      IF Q<5 THEN  GO TO 6230
6310 GO SUB 5900
6320 LET X=2
6330 GO SUB 5800
6399 RETURN
```

La subrutina 5900 lleva a la hormiga Q hasta la posición actual del hombre. Una vez que la hormiga ha alcanzado al hombre, la línea 6240

llama a la subrutina 5600 para localizar la situación de las galerías. Si sólo hay una galería, su posición es guardada en X1, y se asigna Ø a X2. Si hay 2 galerías que conectan un túnel con el de abajo, sus posiciones se almacenan en X1 y X2. Una vez localizadas las galerías, la línea 6260 comprueba cuál es la más cercana a la hormiga que ha capturado al hombre. Después se llama a la subrutina 5800 para mover la hormiga y el hombre a la galería más cercana. Finalmente, la subrutina 5500 es llamada para dejar caer al hombre por la galería. Esta secuencia se repite hasta que el hombre llega al nivel más bajo, y la hormiga 5 arrastra al hombre hasta el nido (línea 6300) ¡¿Fácil, no?!

Por supuesto que la sencillez de la subrutina 6200 se debe al uso de varias subrutinas, la 5900 sólo tiene que mover la hormiga desde su posición actual hasta XM, pero antes de este movimiento la hormiga tiene que apuntar a la dirección adecuada:

```
5900 REM ir a hombre
5905 INK 6
5910 IF A(Q,1)+1=XM
      THEN  RETURN
5920 IF SGN (XM-A(Q,1)-1)<>
        A(Q,3) THEN  GO SUB 3500
5940 PRINT AT A(Q,2),A(Q,1);
     "  ";
5950 LET A(Q,1)=A(Q,1)+A(Q,3)
5960 PRINT AT A(Q,2),A(Q,1);
        A$(Q);
5970 IF A(Q,1)+1<>XM
      THEN  GO TO 5940
5999 RETURN
```

La línea 5920 comprueba si la hormiga mira hacia el hombre, y, si no es así, se llama a la subrutina 3500. Después de esto, el bucle (líneas 5940 a 5970), mueve la hormiga una posición cada vez hasta que toca al hombre.

La subrutina 5600 funciona examinando la fila de posiciones de carácter justo debajo del túnel, y almacena las coordenadas de las galerías en X1 y X2.

```
5600 REM encontrar pozos
5610 LET X=0
5620 LET X1=X
5630 IF X>27 OR
        ATTR (A(Q,2)+1,X)=2
      THEN  GO TO 5660
5640 LET X=X+1
5650 GO TO 5620
5660 LET X2=0
```

```
5670 LET X=X+1
5680 IF ATTR (A(Q,2)+1,X)=2
      THEN  LET X2=X
5690 IF X<27 AND X2=0
      THEN  GO TO 5670
5699 RETURN
```

El primer bucle (líneas 5620 a 5650) busca la primera galería, y el segundo bucle (líneas 5660 a 5690) busca la segunda. Si no hay segunda galería, el segundo bucle acaba cuando llega al final de la pantalla, y devulve X2 con cero.

La tercera nueva subrutina que emplea la subrutina 6200 es la 5800, que es muy parecida a la 5900:

```
5800 REM arrastrar
5810 IF A(Q,1)+2=X
      THEN  LET XM=X:
        PRINT AT A(Q,2),A(Q,1);
        A$(Q);: RETURN
5820 IF SGN (X-A(Q,1)-2)<>A(Q,3)
      THEN  GO SUB 3500
5830 PRINT AT A(Q,2),A(Q,1);
     "   ";: REM tres espacios
5840 LET A(Q,1)=A(Q,1)+A(Q,3)
5850 PRINT AT A(Q,2),A(Q,1);
      INK 6;A$(Q); INK 3;"[E]";
5860 IF A(Q,1)+2<>X
      THEN  GO TO 5830
5870 LET XM=X
5899 RETURN
```

La única diferencia real entre las subrutinas 5800 y 5900 es que la 5800 mueve la hormiga y el hombre hasta la galería más cercana, mientras que la 5900 mueve sólo la hormiga hasta el hombre.

Mejorando el juego

Después de jugar unas cuantas veces, el juego resulta tan simple, que es muy fácil ganar. Un juego demasiado fácil no es, en absoluto, «juego», y ésta puede ser una razón para pensar que el juego es un fracaso. De hecho, la primera versión de la mayor parte de los juegos no resulta casi nunca como se pensaba por una u otra razón, y el siguiente paso es ver en qué falla, y corregirlo.

El primer impulso es intentar hacer el juego más difícil disminuyendo el tiempo límite, pero esto no sirve para nada, por más que se dismi-

nuya, porque llega un momento en que resulta casi imposible jugar. El problema es que el juego siempre dura más o menos el mismo tiempo, y si se aumenta éste, el juego es fácil. ¡¡ Pero, si se disminuye, el juego es imposible!!. Para hacer que el juego sea más interesante, lo primero que hay que hacer es encontrar el motivo de que siempre se requiera el mismo tiempo.

La razón más obvia es que inicialmente todas las hormigas se mueven hacia la derecha, y si el hombre va rápidamente por la izquierda, llega al nido antes de que las hormigas tengan tiempo de dar la vuelta. La solución es hacer que las hormigas se muevan en direcciones aleatorias desde el principio. Puesto que ya tenemos escrita la subrutina 3500, lo mejor es añadir la siguiente línea:

```
1245 IF RND*1>Ø.5
       THEN  GO SUB 35ØØ
```

Después de este cambio, el juego es más difícil y más interesante, pero aún es posible que el hombre se mueva entre las hormigas sin ser capturado, y esto querrá decir que no tiene que preocuparse de los lugares en donde están las hormigas. Esto se debe a que la comprobación de la captura del hombre sólo se hace cuando una hormiga dada se mueve, y la hormiga se mueve una vez cada 5 que el hombre lo hace. La solución es añadir una línea para comprobar si el hombre llega a una posición ya ocupada por una hormiga:

```
5075 GOSUB 5100
```

y después:

```
51ØØ REM capturan las hormigas
5110 IF YM<3 OR ATTR (YM,XM)<>6
       THEN  RETURN
5120 LET HIT=1
5130 LET Q=1
5140 IF A(Q,2)=YM THEN  RETURN
5150 LET Q=Q+1
5160 IF Q<5 THEN  GO TO 5140
5199 RETURN
```

Esto hace la comprobación de si el hombre ha llegado a una hormiga, y, si es así, hace los cálculos necesarios.

Una vez hecho este cambio, otra característica fastidiosa se hace evidente. Como una hormiga puede moverse de tres en tres posiciones, es posible que ésta «salte» sobre el hombre. Para que esto no suceda, basta con restringir el movimiento de las hormigas a un máximo de dos posiciones cada vez. Para escribir esto, basta con cambiar la línea 3050 de la subrutina 3000 por:

```
3050 LET A(Q,1)=A(Q,1)+A(Q,3)*2
```

Así, todas las hormigas se mueven constantemente de dos en dos espacios. Después del cambio parece que dichas hormigas se mueven lo bastante rápido y variado, para que el juego parezca interesante.

Ya están hechos los cambios más obvios, y el juego es más difícil y divertido, pero no tiene el grado de «reto» que todo buen juego ha de tener. Después de jugar varias veces, nos damos cuenta de que se debe a que las hormigas están demasiado tiempo lejos de las galerías, y no suponen ninguna amenaza para el hombre.

Los mejores juegos son aquellos en los que, por suerte, todas las hormigas están cerca del centro de la pantalla. Esto sugiere que el área de movimiento de las hormigas debe ser reducida, y los pares de galerías deben estar más cercanos. Para ello las líneas 3060 y 3070 de la subrutina 3000 han de ser cambiadas por:

```
3050 LET A(Q,1)=A(Q,1)+A(Q,3)*2
3060 IF A(Q,1)<10
       THEN  LET A(Q,1)=10:
         GO SUB 3500
3070 IF A(Q,1)>25
       THEN  LET A(Q,1)=25:
         GO SUB 3500
```

y las líneas 9010 y 9020 de la 9000 por:

```
9000 REM pozos
9010 LET X1=INT (RND*3)+10
9020 LET X2=INT (RND*5)+18
9030 FOR Y=Y1 TO Y2
9040 IF NUM=1 THEN
```

Con estos cambios, «La colina de las hormigas» combina velocidad y estrategia, y es un buen juego.

Sugerencias para posteriores trabajos

La mayor parte del juego está bastante completo, y el mejor campo para trabajos posteriores está constituido por las rutinas del final del juego. Las subrutinas 6500 y 6800 son bastante «flojas» comparadas con la forma de moverse las hormigas al llamar a la subrutina 6200.

La versión final - Listado completo.

El siguiente listado incluye todas las modificaciones introducidas en el texto:

```
 10 REM el monte de la hormiga
 20 CLS
 30 GO SUB 1000
 40 GO SUB 8000
 50 GO SUB 2000
 60 REM bucle
 70 GO SUB 4000
 80 IF TIMEUP OR HIT OR HOME
     THEN  GO TO 100
 90 GO TO 60
100 GO SUB 6000
110 CLS
199 STOP


1000 REM inicializacion
1010 FOR Q=97 TO 105
1020 FOR R=0 TO 7
1030 READ D
1040 POKE USR (CHR$ (Q))+R,D
1050 NEXT R
1060 NEXT Q
1070 REM datos
1080 DATA 0,14,15,63,31,51,68,
          132
1090 DATA 3,20,216,252,168,192,
          64,16
1100 DATA 0,112,248,252,248,204,
          34,33
1110 DATA 192,40,27,63,25,3,4,8
1120 DATA 24,24,126,24,60,102,
          195,195
1130 DATA 1,1,3,3,7,7,15,15
1140 DATA 31,31,63,63,127,127,
          255,255
1150 DATA 128,128,192,192,224,
          224,240,240
1160 DATA 248,248,252,252,254,
          254,255,255
1170 LET XM=INT (RND*20)+10
1180 LET YM=1
1190 DIM A(5,3): DIM A$(5,2)
1200 FOR Q=1 TO 5
1210 LET A(Q,1)=INT (RND*25)+5
1220 LET A(Q,2)=Q*4+1
```

```
1230 LET A(Q,3)=1
1240 LET A$(Q)="   "
1245 IF RND*1>0.5
      THEN  GO SUB 3500
1250 NEXT Q
1260 LET HIT=0
1270 LET HOME=0
1280 LET TIMEUP=0
1299 RETURN

2000 REM dibujo del hormiguero
2010 PAPER 3
2020 BORDER 3
2030 CLS
2040 PAPER 0
2050 FOR X=0 TO 31
2060 PRINT AT 5,X;" "
2070 PRINT AT 9,X;" "
2080 PRINT AT 13,X;" "
2090 PRINT AT 17,X;" "
2100 PRINT AT 21,X;" "
2110 NEXT X
2120 PAPER 5
2130 FOR X=0 TO 31
2140 FOR Y=0 TO 1
2150 PRINT AT Y,X;" ";
2160 NEXT Y
2170 NEXT X
2180 PAPER 0
2190 FOR I=1 TO 5
2200 READ NUM,Y1,Y2
2230 GO SUB 9000
2240 NEXT I
2250 DATA 2,2,4
2260 DATA 1,6,8
2270 DATA 2,10,12
2280 DATA 1,14,16
2290 DATA 2,18,20
2300 INK 3: PAPER 5
2310 PRINT AT YM,XM;" ";
2320 INK 0: PAPER 3
2330 PRINT AT 20,1;"   "
2340 PRINT AT 21,1;"   "
2350 PAPER 0
2360 FOR I=1 TO 5
```

```
2370 PRINT AT A(I,2),A(I,1);
      INK 6;A$(I);
2380 NEXT I
2390 POKE 23672,0: POKE 23673,0:
      POKE 23674,0
2399 RETURN


3000 REM  mover la hormiga
3010 PAPER 0
3020 LET Q=INT (RND*5)+1
3030 PRINT AT A(Q,2),A(Q,1);
     "  ";
3040 IF RND*1<.01
      THEN  GO SUB 3500
3050 LET A(Q,1)=A(Q,1)+A(Q,3)*2
3060 IF A(Q,1)<10
      THEN  LET A(Q,1)=10:
        GO SUB 3500
3070 IF A(Q,1)>25
      THEN  LET A(Q,1)=25:
        GO SUB 3500
3080 PRINT AT A(Q,2),A(Q,1);
      INK 6;A$(Q)
3085 INK 7
3090 GO SUB 9200
3099 RETURN


3500 REM cambio de sentido
3510 IF A$(Q)="[AB]"
      THEN  LET A$(Q)="[DC]":
        GO TO 3530
3520 LET A$(Q)="[AB]"
3530 LET A(Q,3)=-A(Q,3)
3599 RETURN


4000 REM un movimiento
4010 GO SUB 5000
4015 IF HIT=1 THEN  GO TO 4099
4020 GO SUB 3000
4030 GO SUB 4500
4099 RETURN
```

```
4500 REM tiempo
4510 LET T=(PEEK 23672+
          256*PEEK 23673+
          65536*PEEK 23674)/100
4520 PRINT AT 0,27;INT (T);
4530 IF T>MAX THEN  LET TIMEUP=1
4599 RETURN


5000 REM mover el hombre
5010 PAPER 0
5020 IF YM<2 THEN  PAPER 5
5030 PRINT AT YM,XM;" ";
5040 IF INKEY$="5" AND XM>0
      THEN  LET XM=XM-1
5050 IF INKEY$="8" AND XM<31
      THEN  LET XM=XM+1
5060 IF INKEY$="6" AND
       ATTR (YM+1,XM)=2
      THEN  GO SUB 5500
5070 PRINT AT YM,XM; INK 3;
     "[E]";
5075 GO SUB 5100
5080 IF XM=3 AND YM=21
      THEN  LET HOME=1
5099 RETURN


5100 REM capturan las hormigas
5110 IF YM<3 OR ATTR (YM,XM)<>6
      THEN  RETURN
5120 LET HIT=1
5130 LET Q=1
5140 IF A(Q,2)=YM THEN  RETURN
5150 LET Q=Q+1
5160 IF Q<5 THEN  GO TO 5140
5199 RETURN


5500 REM bajar por el pozo
5510 PAPER 0: INK 3
5520 FOR I=1 TO 4
5530 LET YM=YM+1
5540 PRINT AT YM,XM;"[E]";
5550 BEEP .075,0-(I*4+10)
```

```
5560 PRINT AT YM,XM;" ";
5570 NEXT I
5599 RETURN


5600 REM encontrar pozos
5610 LET X=0
5620 LET X1=X
5630 IF X>27 OR
        ATTR (A(Q,2)+1,X)=2
      THEN  GO TO 5660
5640 LET X=X+1
5650 GO TO 5620
5660 LET X2=0
5670 LET X=X+1
5680 IF ATTR (A(Q,2)+1,X)=2
      THEN  LET X2=X
5690 IF X<27 AND X2=0
      THEN  GO TO 5670
5699 RETURN


5800 REM arrastrar
5810 IF A(Q,1)+2=X
      THEN  LET XM=X:
        PRINT AT A(Q,2),A(Q,1);
        A$(Q);: RETURN
5820 IF SGN (X-A(Q,1)-2)<>A(Q,3)
      THEN  GO SUB 3500
5830 PRINT AT A(Q,2),A(Q,1);
     "   ";: REM tres espacios
5840 LET A(Q,1)=A(Q,1)+A(Q,3)
5850 PRINT AT A(Q,2),A(Q,1);
      INK 6;A$(Q); INK 3;"[E]";
5860 IF A(Q,1)+2<>X
      THEN  GO TO 5830
5870 LET XM=X
5899 RETURN


5900 REM ir a hombre
5905 INK 6
5910 IF A(Q,1)+1=XM
      THEN  RETURN
5920 IF SGN (XM-A(Q,1)-1)<>
```

```
        A(Q,3) THEN  GO SUB 3500
5940 PRINT AT A(Q,2),A(Q,1);
     "  ";
5950 LET A(Q,1)=A(Q,1)+A(Q,3)
5960 PRINT AT A(Q,2),A(Q,1);
          A$(Q);
5970 IF A(Q,1)+1<>XM
      THEN  GO TO 5940
5999 RETURN


6000 REM final del juego
6010 IF HIT THEN  GO SUB 6200:
       GO TO 6100
6020 IF TIMEUP THEN  GO SUB 6500
     : GO TO 6100
6030 IF HOME THEN  GO SUB 6800
6100 FOR I=1 TO 100: NEXT I
6105 PAPER 7
6110 INK 0
6120 BORDER 7
6130 CLS
6140 INPUT "Otra partida s/n?";
     N$
6150 IF N$="s" THEN  RUN
6160 IF n$<>"n" THEN  GO TO 6140
6199 RETURN


6200 REM captura
6210 PRINT AT 0,10;
     "HAS SIDO CAPTURADO"
6220 IF Q=5 THEN  GO TO 6320
6230 GO SUB 5900
6240 GO SUB 5600
6250 LET X=X1
6260 IF ABS (X2-A(Q,1))<
        ABS (X1-A(Q,1))
      THEN  LET X=X2
6270 GO SUB 5800
6280 PRINT AT A(Q,2),A(Q,1)+2;
     " ";
6290 BEEP .1,-10
6295 GO SUB 5500
6296 PRINT AT YM,XM;"[E]";
```

```
6300 LET Q=Q+1:
      IF Q<5 THEN  GO TO 6230
6310 GO SUB 5900
6320 LET X=2
6330 GO SUB 5800
6399 RETURN


6500 REM salir del cascaron
6510 PRINT AT 21,1;
     "LAS HORMIGAS SALEN DEL ";
     "CASCARON"
6520 FOR I=1 TO RND*20+20
6530 LET X=RND*25+5
6540 LET Y=RND*17+3
6550 PRINT AT Y,X; FLASH 1;
     A$(INT (RND*5)+1);
6560 BEEP .1,RND*20-30
6570 NEXT I
6599 RETURN


6800 REM destruir nido
6805 RESTORE 6820
6810 PRINT AT 21,1;
     "LO CONSEGUISTE"
6820 DATA 5,.25,7,.25,8,.25,11,
     .25,12,.5,12,.5
6830 DATA 999,999
6840 READ P,D
6850 IF P=999 THEN  GO TO 6899
6860 BEEP D,P
6870 GO TO 6840
6899 RETURN


8000 REM titulo
8010 INK 2: PAPER 6: CLS
8020 PRINT AT 3,6;
     "MONTE DE LA HORMIGA"
8030 FOR I=1 TO 16: PRINT A$(1):
      NEXT I
8040 PRINT AT 5,2;
     "En este juego tienes un ";
     "tiempo"
```

```
8050 PRINT AT 6,2;"fijo para ";
     "destruir el nido de "
8060 PRINT AT 7,2;"hormigas ";
     "antes de que las"
8070 PRINT AT 8,2;"crias ";
     "salgan del cascaron!"
8080 PRINT AT 9,2;"Usa los ";
     "cursores para moverte";
     AT 10,2;"pero"
8090 PRINT AT 11,2;"CUIDADO:";
     "las hormigas soldado"
8100 PRINT AT 12,2;"que ";
     "guardan los tuneles te "
8105 PRINT AT 13,2;"apresaran";
     " y te llevaran al"
8110 PRINT AT 14,2;"nido para ";
     ."alimentar a sus ";
     AT 15,2;"crias"
8120 PRINT AT 16,2;"Buena ";
     "suerte"
8200 PRINT AT 18,2;"Nivel de ";
     "dificultad_"
8210 PRINT AT 19,2;"1.Experto,";
     " 2.Medio, 3.novato ?";
8220 INPUT DF
8230 IF DF<1 OR DF>3
      THEN  GO TO 8200
8240 LET MAX=(5+DF*20)+
     INT (RND*10)
8299 RETURN


9000 REM pozos
9010 LET X1=INT (RND*3)+10
9020 LET X2=INT (RND*5)+18
9030 FOR Y=Y1 TO Y2
9040 IF NUM=1 THEN
      PRINT AT Y,((X1+X2)/2);
      " ";
9050 IF NUM=2 THEN
      PRINT AT Y,X1;" ";AT Y,X2;
      " ";
9060 NEXT Y
9099 RETURN
```

```
9200 REM atraparon al hombre
9210 IF A(Q,2)<>YM THEN  RETURN
9220 IF A(Q,1)<>XM AND
     A(Q,1)+1<>XM THEN  RETURN
9230 LET HIT=1
9299 RETURN
```

3. El salto de la rana

El salto de la rana es un juego animado en 2 dimensiones. En él, uno de los personajes, la rana, se mueve al mismo tiempo horizontal y verticalmente (mientras que en el anterior capítulo el movimiento era bien vertical, bien horizontal). Además se utilizan interesantes técnicas de animación para crear en la pantalla unas imágenes llamativas, y que no necesitan demasiadas sentencias BASIC. El juego ofrece muchas posibilidades para variaciones, y es muy divertido no sólo para jugar, sino también para verlo simplemente.

Diseño del juego

El salto de la rana utiliza muchos de los elementos que se pueden encontrar en uno de los primeros juegos de computadoras, que se conoce por varios nombres; «Break Out», «El Muro»... Estos juegos consisten en hacer botar una pelota alrededor de la pantalla por medio de una raqueta, con el objetivo de borrar el mayor número posible de bloques coloreados o ladrillos, que aparecen en una banda en la parte superior de la pantalla.

En las versiones más simples, se trata de mover la raqueta para interceptar la pelota y, y así, conseguir que siga botando. En las más sofisticadas, el jugador puede dirigir la pelota, según con que parte de la raqueta se golpee ésta.

En el salto de la rana, los bloques de colores o ladrillos son sustituidos por hileras de insectos en formación, y la pelota es reemplazada por una rana saltarina. Obviamente, resultaría un poco cruel utilizar una simple raqueta para conseguir que la rana no pare de moverse, y es por lo que, en lugar de la raqueta, se utiliza un «balancín». Una segunda rana se sienta en el balancín, y cuando la primera rana aterriza, la segunda es catapultada hacia el aire, y la primera rana se queda sentada en el balancín, esperando su turno.

Otra diferencia entre el tradicional juego de «El Muro» y el salto de la rana, es la trayectoria que la rana describe en el aire. En «El Muro» la pelota se mueve en líneas rectas en sus botes alrededor de la pantalla, pero en «El Salto de la Rana», ésta se mueve como si fuese atraída hacia la tierra por el efecto gravedad.

Cada vez que la rana aterriza en el balancín, la rana que es catapultada hacia el aire, bota un poco más alto; por tanto, para alcanzar la última hilera de insectos, es necesario conseguir que las ranas no paren de moverse.

Esto es básicamente el juego del salto de la rana, pero, antes de escribirlo en BASIC, es útil revisar la forma de conseguir que un «sprite» bote alrededor de la pantalla, y se mueva bajo los efectos de una gravedad simulada.

La figura 3. 1 muestra la imagen de la pantalla cuando el programa se ejecuta.

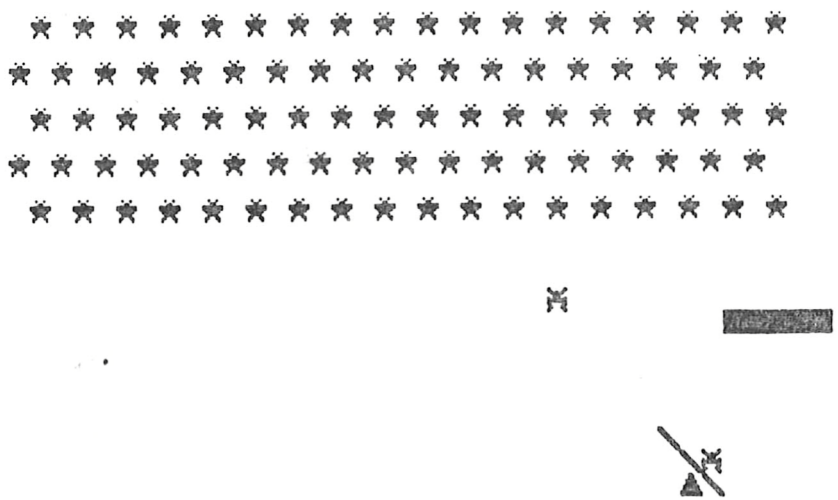


Fig. 3.1

Botando y cayendo

En el capítulo 2 ya ha sido descrita la idea de animar un objeto, sin más que actualizar repetidamente sus coordenadas de posición, para lo que se añaden las cantidades que corresponden a las velocidades horizontal y vertical.

El ciclo básico de la animación es:

```
PRINT AT Y,X;" ";:REM Borrar objeto
LET X=X+VX:LET Y=Y+VY:REM Actualizar coordenadas
```

```
PRINT AT Y,X;C$;:REM Reimprimir objeto
```

donde X e Y indican la posición del objeto y VX e VY indican, respectivamente, la componente horizontal y vertical de la velocidad.

Utilizando este esquema, el objeto aparecerá trasladándose de una parte a otra de la pantalla, siguiendo la misma línea de movimiento hasta que desaparece.

Esta clase de movimiento no es muy útil en un juego, y es por lo que aparece la necesidad de hacer botar el objeto. En principio, botar un objeto contra un límite horizontal o vertical es fácil. Si un «sprite» que se mueve con velocidad VX, VY choca contra el límite horizontal, cambia su velocidad por VX, -VY, es decir, un bote contra un límite horizontal invierte la velocidad vertical. Del mismo modo, un bote contra un límite vertical invierte la velocidad horizontal.

Como un ejemplo, tratamos de seguir el siguiente programa:

```
10 LET VX=1
20 LET VY=1
30 LET X=RND*31: LET Y=RND*21
40 PRINT AT Y,X;" ";
50 LET X=X+VX: LET Y=Y+VY
60 IF X<1 OR X>30 THEN LET VX=-VX
70 IF Y<1 OR Y>21 THEN LET VY=-VY
80 PRINT AT Y,X;"A";.
90 GO TO 40
```

La línea 60 detecta las colisiones con los bordes verticales de la pantalla, de modo que se invierte la velocidad horizontal. La línea 70 detecta las colisiones con el borde horizontal de la pantalla, y entonces invierte la componente vertical de la velocidad. El resultado es una letra «A» que bota alrededor de la pantalla. Resulta sorprendente que este tipo de animación necesite tan pocas líneas de BASIC.

Cualquier otro tipo de movimiento más complicado se puede conseguir sin más que cambiar cada vez las velocidades horizontal y vertical dentro del bucle de animación. Por ejemplo, para hacer que un «sprite» se mueva a través de la pantalla como si estuviera bajo la influencia de la gravedad, no hay más que cambiar la velocidad vertical por una cantidad fija, cada vez dentro del bucle.

Esto es lo que sucede cuando una pelota real es lanzada al aire. La pelota empieza su movimiento con una cierta velocidad vertical, que se va reduciendo constantemente por causa de la gravedad hasta que llega a cero, y entonces cambia de dirección haciendo caer la pelota hacia el suelo cada vez más rápido. La gravedad no afecta a la velocidad horizontal de la pelota. Los únicos factores que la modifican, son la resistencia del aire o viento, y en la mayor parte de los casos se pueden ignorar.

El programa siguiente consigue que el «sprite» se mueva como una pelota lanzada al aire:

```
10 CLS
20 LET X=0: LET Y=21
30 LET VX=.3: LET VY=-.5
40 LET A=.01
50 PRINT AT Y,X;"  ";
60 LET X=X+XV: LET Y=Y+YV
70 LET VY=VY+A
80 PRINT AT Y,X;"A";
90 GO TO 50
```

La línea 60 realiza la actualización usual de las coordenadas, pero en este programa la línea 70 también actualiza la velocidad vertical añadiéndole una constante.

El resultado es una letra «A» que describe una parábola, y desaparece por la parte inferior de la pantalla. La magnitud de la constante utilizada en la línea 40 controla la fuerza de la gravedad a la cual la letra «A» responde. Si la aumentamos, la letra «A» caerá más ràpido.

Hay que observar que, para tener en cuenta el efecto de la gravedad, es necesario utilizar coordenadas que no son enteras, y esto puede causar complicaciones posteriores..

El programa principal

Después de esta discusiones teóricas, volvamos a los detalles del programa del «salto de la rana». Veamos los elementos del juego: El jugador dispone de 10 ranas para intentar comerse todos los insectos situados en la parte superior de la pantalla. Cada nueva rana empieza saltando desde la plataforma, y el usuario tiene que colocar el balancín, para catapultar a la segunda rana hacia el aire.

Una vez que se ha comenzado de esta manera, el bucle principal de animación sigue hasta que el jugador «pierde» una rana con el balancín. Una nueva rana aparece entonces, y el juego continúa.

Hay dos formas de terminar el juego, bien porque el jugador consigue comerse todos los insectos, o bien porque se acaban las 10 ranas. El programa principal es:

```
10 REM salto de rana
30 GO SUB 1000
40 GO SUB 6000
50 GO SUB 2000
60 GO SUB 5000
```

```
 7Ø REM bucle
 9Ø GO SUB 3ØØØ
1ØØ GO SUB 4ØØØ
11Ø IF GAMEEND<>1
     THEN  GO TO 7Ø
12Ø GO SUB 9ØØØ
13Ø IF AGAIN THEN  RUN
199 STOP
```

La subrutina 1000 establece los caracteres definidos por el usuario, y define las otras constantes más tarde utilizadas en el programa. La subrutina 6000 imprime una pantalla de intrucciones, y pide al usuario que elija el nivel de dificultad.

La subrutina 2000 imprime 5 filas de insectos en la parte superior de la pantalla, listos para que el juego empiece.

La subrutina 5000 hace que una rana salte desde la plataforma, mientras que el jugador intenta colocar el balancín debajo de ella.

El bucle principal de animación está formado por las líneas 70 hasta 110.

La subrutina 3000 permite al jugador mover el balancín utilizando las teclas de las flechas derecha e izquierda, y la 4000 mueve la rana teniendo en cuenta la gravedad y cualquier bote que sea necesario. Finalmente, la subrutina 9000 controla la puntuación del jugador, y le pregunta si quiere volver a jugar.

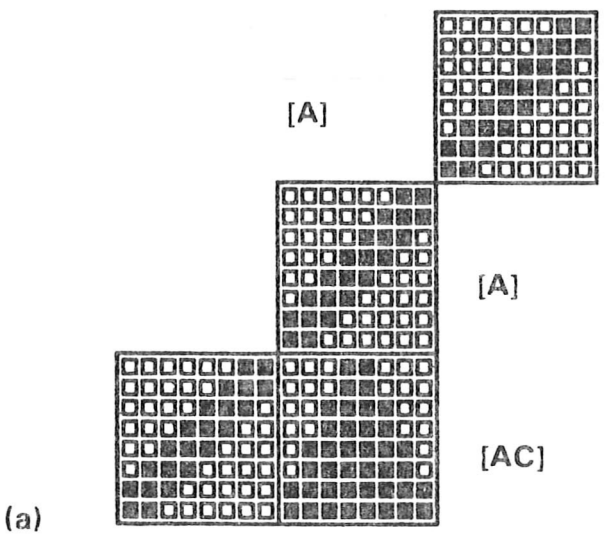
Subrutina 1000 y subrutina 6000

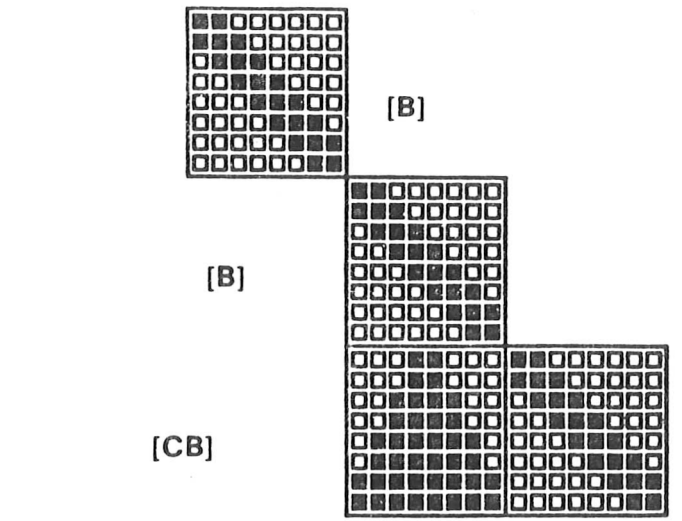
La sección central de la subrutina 1000 parece un poco complicada, pero la verdad es que es una subrutina bastante fácil de comprender:

```
1ØØØ REM inicio
1Ø1Ø DIM F$(2)
11ØØ DATA "A",3,7,14,28,56,112,
     224,192
1110 DATA "B",192,224,112,56,28,
     14,7,3
1120 DATA "C",24,24,6Ø,6Ø,126,
     126,255,255
1130 DATA "D",36,24,255,255,126,
     6Ø,1Ø2,66
1140 DATA "E",13Ø,84,56,186,254,
     13Ø,68,13Ø
```

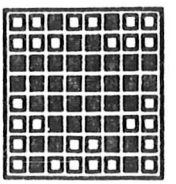
```
1210 FOR I=1 TO 5
1220 READ C$
1230 FOR J=0 TO 7
1240 READ D
1250 POKE USR C$+J,D
1260 NEXT J
1270 NEXT I
1300 DIM L$(70)
1310 DIM R$(70)
1320 LET L$="[    A                ]"
     +"[                       A    ]"
     +"[                            ]"
     +"[      AC]"
1330 LET R$="[ B                    ]"
     +"[                       B    ]"
     +"[                            ]"
     +"[       CB]"
1400 LET M=1
1410 LET S$=L$
1420 LET X=21
1430 LET FROG=1
1440 LET GAMEEND=0
1450 LET SC=0
1499 RETURN
```

Las líneas 1100 a 1270 definen las cinco figuras empleadas en el juego, que están ilustradas en la Figura 3. 2 «[A]» y «[C]» se unen para formar el «balancín izquierda», y «[B]» y «[C]» se unen para formar el «balancín derecha», «[B]» es la figura del insecto, y «[E]» es la figura de la rana.

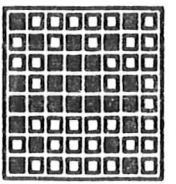




(b) Right Seesaw



(c)



(d)

Fig. 3.2

Caracteres gráficos para: a) Balancín izquierda; b) Balancín derecha; c) Insecto, y d) Rana.

Las líneas 1300 a 1330 definen dos matrices L$ y R$, que contienen la secuencia de caracteres que imprimen los balancines izquierda y derecha respectivamente. Las líneas 1400 a 1450 definen los valores iniciales de varias variables que se utilizarán en el resto del programa. M se utiliza para indicar qué balancín se está empleando. Si M es 1, es el balancín izquierda, y si M es Ø, es el derecha. S$ es la cadena que guarda los caracteres que produce el balancín en uso. Es decir, PRINT AT Y, X; S$ imprime la forma del balancín actual con la esquina superior izquierda en X, Y. X da la posición horizontal del balancín cuando empieza el juego. GAMMEND se usa para señalar el final del juego, y SC almacena la puntuación actual.

La subrutina 6000 simplemente imprime las instrucciones del juego, pregunta qué nivel de dificultad se desea, y lo guarda en D:

```
6ØØØ REM titulo
6Ø1Ø PAPER 6
6Ø2Ø CLS
6Ø3Ø INK 4
6Ø4Ø PRINT AT 2,3;
     "S A L T O  ";
     " D E   R A N A"
6Ø5Ø INK Ø
6Ø6Ø PRINT AT 6,Ø;
     "En este juego debes ";
     "comerte las"
6Ø7Ø PRINT "moscas [D]"
6Ø8Ø PRINT
6Ø9Ø PRINT "haciendo saltar ";
     "a tus dos""ranas [E]"
61ØØ PRINT "del balancin,";
     " usando las teclas"
611Ø PRINT "de cursor ";
     "izquierdo y derecho."
612Ø PRINT "Tienes diez ranas";
     " para limpiar"
613Ø PRINT "la pantalla."
614Ø PRINT "Elige el nivel de";
     " dificultad"
615Ø INPUT "1 (dificil) A ";
     "1Ø (facil)";DF
616Ø IF DF<1 OR DF>1Ø THEN
      GO TO 615Ø
6199 RETURN
```

Subrutina 2000

La subrutina 2000 es muy fácil, y se encarga de imprimir cinco filas de insectos:

```
2000 REM imprimir las moscas
2010 INK 6
2020 PAPER 6
2030 BORDER 6
2040 CLS
2050 LET C=0
2060 FOR V=2 TO 10 STEP 2
2070 LET C=NOT C
2080 FOR U=2 TO 26 STEP 2
2090 PRINT AT V,U-C; INK 1;
     "[D]";
2100 NEXT U
2110 NEXT V
2199 RETURN
```

La única técnica interesante es el empleo de C en la línea 2070 para determinar la separación de las filas. Para entender cómo funciona, basta con saber que si C = 0, entonces NOT C es -1.

La tinta es del mismo color que el papel (amarillo) al principio de la subrutina, línea 2010. Cuando en la línea 2090 se imprimen las moscas, se utiliza temporalmente el color uno, es decir azul. Esto es para que en posteriores comprobaciones de los atributos de color de la pantalla en la posición de la rana, se pueda diferenciar entre las moscas y el fondo.

Subrutina 5000

Después de la subrutina 2000, la 5000 imprime una rana sobre una plataforma en la parte de la derecha de la pantalla, y la lleva firmemente hasta el borde de dicha plataforma. Cuando la rana alcanza el borde, salta hacia abajo. Si el balancín está debajo de la rana que cae, el juego continúa, si no es así, la subrutina 5000 es llamada otra vez desde dentro del bucle de animación:

```
5000 REM arranque
5010 GO SUB 3000
5020 INK 0: PAPER 0
5030 PRINT AT 15,26;"[      ]":
     REM graficos shift 8
5040 INK 4: PAPER 6
5050 FOR F=30 TO 24 STEP -.1
```

```
5060 PRINT AT 14,F;"[E ]";
5070 GO SUB 3000
5080 NEXT F
5090 FOR G=14 TO 18 STEP .5
5100 PRINT AT G-1,24;" ";
5110 PRINT AT G,24;"[E]";
5120 GO SUB 3000
5130 NEXT G
5140 LET F=24
5150 LET G=18
5155 LET C=RND*1
5160 LET A=.4+C/10
5170 LET B=-(.1+C/2.5)
5180 LET P=.008
5190 PRINT AT 0,10;"rana ";FROG;
     " tanteo=";SC
5200 LET A=A+A/DF
5210 LET B=B+B/DF
5220 LET P=P+P/DF
5230 GO SUB 4500
5299 RETURN
```

La línea 5010 llama a la subrutina 3000, que en este caso sirve para dibujar el balancín en su posición inicial. Luego la línea 5030 dibuja una pequeña línea de blancos como plataforma, para que la rana salte. El siguiente bucle FOR, en las líneas 5050 a 5080 mueve la rana hasta el borde. Cada vez dentro del bucle, la subrutina 3000 es llamada por la línea 5070, para dar al jugador una oportunidad de mover el balancín. El segundo bucle FOR, líneas 5090 a 5130, mueve la rana verticalmente hacia abajo.

Las líneas 5140 a 5220 inicializan las variables. F y G almacenan la posición actual de la rana, es decir la rana está AT F, G. Las variables A y B son las velocidades vertical y horizontal de la rana. Ambas tienen asignados valores aleatorios (por las líneas 5160 y 5170 respectivamente). La línea 5180 asigna 0.008 a P «constante de gravitación». Las líneas 520 y 5220 ajustan las velocidades y la constante de gravitación, teniendo en cuenta el nivel de dificultad elegido por el jugador. Cuando DF es alto, aumentan las velocidades vertical y horizontal, y la rana se mueve más deprisa. Por último, la línea 5230 llama a la subrutina 4500, para comprobar si la rana ha aterrizado en el balancín o no.

Subrutinas 3000 y 4000

La subrutina 3000 permite al jugador mover el balancín por medio de las flechas. Imprimir el balancín es muy sencillo:

```
3000 REM mover el balancin
3010 IF INKEY$="5" AND X>0 THEN
      LET X=X-1
3020 IF INKEY$="8" AND X<25 THEN
      LET X=X+1
3040 PRINT AT 19,X; INK 2;S$;
3050 LET K=X: LET F$="[ E]"
3060 IF M=0 THEN  LET K=X+3:
       LET F$="[E ]"
3070 INK 4
3080 PRINT AT 20,K;F$;
3099 RETURN
```

Las líneas 3010 y 3020 controlan las flechas derecha e izquierda y actualizan la posición horizontal del balancín (almacenada en X). La línea 3030 lo imprime en su nueva posición. Hay que observar que no es necesario borrar explícitamente el antiguo balancín, debido a los blancos incluidos en cada final de balancín. Las líneas 3040 a 3070 imprimen la segunda rana en el lado más bajo del balancín , preparada para ser catapultada hacia el aire. Las líneas 3050 a 3060 establecen la posición horizontal de esta rana, teniendo en cuenta qué versión (derecha o izquierda) del balancín se necesita.

La subrutina 4000 viene rodada, y sigue la teoría de botar objetos dada al principio del capítulo:

```
4000 REM mover la rana
4010 PRINT AT G,F; PAPER 8;" ";
4020 LET F=F+B
4030 LET G=G+A
4040 LET A=A+P
4050 IF F>27 OR F<2 THEN
      LET B=-B
4060 IF G<2 THEN  LET G=2:
       LET A=-A
4070 IF G<11 THEN  GO SUB 7000
4080 PRINT AT G,F; PAPER 8;
     "[E]";
4090 IF G>18 THEN  GO SUB 4500
4099 RETURN
```

La línea 4010 borra la rana de su antigua posición. Las líneas 4020 a 4040 actualizan la posición y la velocidad (recordemos que la posición horizontal está guardada en F, la velocidad vertical en A, y la constante de gravitación en P). La línea 4070 comprueba si la rana está lo bastante alto como para haber comido una rana. La línea 4080 reimprime la rana, y la línea 4090 comprueba si la rana está lo bastante bajo como para haber aterrizado en el balancín, si es así, se llama a la subrutina 4500.

Subrutinas 4500, 8000 y 7000

La subrutina 4500 comprueba si la rana aterriza en el balancín:

```
4500 REM balancin
4505 PRINT AT G,F; PAPER 8;" ";
4510 LET D=INT (F)-X-M
4520 IF D<0 OR D>2 THEN
      GO SUB 8000: RETURN
4550 IF M=1 THEN  GO TO 4590
4560 LET M=1
4570 LET S$=L$
4580 GO TO 4610
4590 LET M=0
4600 LET S$=R$
4610 LET A=-A
4620 LET G=18
4630 LET F=K+1
4640 BEEP .1,-15
4699 RETURN
```

En este caso, las líneas 4530 y 4540 no se escriben, porque se insertan más tarde.

Como la posición horizontal de la rana es F, y la posición horizontal de la esquina superior izquierda del balancín es X, se puede utilizar la diferencia entre ambas para comprobar si la rana ha aterrizado en el balancín o no. El único problema es que el lado del balancín en el que aterriza la rana, depende de si es un balancín izquierda o derecha. Si es izquierda, la diferencia tiene que ser 2 ó 3 para un aterrizaje correcto. Si es derecha, la diferencia tiene que ser 1 ó 2. Restando la variable M, que es 1 para un balancín izquierda, y 0 en caso contrario, se consigue que la diferencia sea 1 ó 2, para el aterrizaje correcto.

Esta diferencia «corregida» es calculada por la línea 4510, y la 4520, que llama a la subrutina 8000 si la rana no ha caído en el balancín. Como la función INT del Spectrum redondea al menor entero, para comprobar las posiciones del balancín, lo haremos desde 0 hasta 2. Si la rana ha aterrizado con éxito, las líneas 4550 a 4600 cambian un tipo de balancín por otro. La línea 4610 invierte la velocidad vertical de la rana cuando ésta bota en el balancín. Las líneas 4620 y 4630 actualizan las coordenadas de la rana del otro lado del balancín.

Si la rana falla, se llama a la subrutina 8000, para que aparezca una nueva, a menos, claro está, que ya se hayan usado las diez:

```
8000 REM error
8010 PRINT AT G,F; PAPER 8;" "
```

```
8020 LET FROG=FROG+1
8030 IF FROG>10 THEN
      LET GAMEEND=1: RETURN
8040 BEEP .1,0
8050 IF M=0 THEN  LET M=1:
       LET S$=L$
8060 GO SUB 5000
8099 RETURN
```

Si ya se han perdido diez ranas, la línea 8030 asigna 1 (VERDAD) a GAMEEND, y llegamos al final. En caso contrario se cambia la versión del balancín (línea 8050), y después la línea 8060 llama a la subrutina 5000 para seguir el juego con una nueva rana.

La subrutina 7000 utiliza ATTR para ver el color del carácter al que va a moverse la rana. Si es azul, la rana ha chocado contra una mosca.

```
7000 REM vuelo de busqueda
7010 IF ATTR (G,F)<>49 THEN
      RETURN
7020 BEEP .1,-20
7030 LET SC=SC+1
7040 PRINT AT 0,10;"rana ";FROG;
     " tanteo=";SC
7050 IF SC=65 THEN
      LET GAMEEND=1
7099 RETURN
```

Si es así, se actualiza la puntuación (línea 7030), y se comprueba si todas las moscas han sido comidas (línea 7050).

Subrutina 9000

Esta subrutina imprime varios mensajes, dependiendo de la puntuación final, y pregunta al jugador si quiere volver a jugar.

```
9000 REM fin de juego
9010 INK 0
9020 CLS
9030 PRINT AT 10,1;
9040 IF SC<10 THEN
      PRINT "Hasta un renacuajo"
      '" lo haria mejor":
       GO TO 9400
9050 IF SC<20 THEN
      PRINT "Tenian tres ";
      "patas tus ranas ?":
```

```
       GO TO 9400
9060 IF SC<40 THEN
      PRINT "No esta mal para";
      " un sapo miope":
       GO TO 9400
9070 IF SC<50 THEN
      PRINT "Bastante bueno ";
      "considerando....":
       GO TO 9400
9080 IF SC<60 THEN
      PRINT "Una gran operacion"
      ;" de vuelo"'" comiendo":
       GO TO 9400
9090 PRINT "Intentalo metiendo";
     " a las"'" ranas olimpicas"
9400 PRINT AT 15,5;"Tu tanteo ";
     SC
9410 PRINT AT 16,1;"Otro juego";
     "(s/n) ?": INPUT  LINE A$
9420 IF A$(1)<>"n" AND
     A$(1)<>"s" THEN  GO TO 9410
9430 IF A$(1)="s" THEN
      LET AGAIN=1
9440 IF A$(1)="n" THEN
      LET AGAIN=0
9499 RETURN
```

Evaluación y mejoras

La versión actual de «El salto de la rana», es bastante rápida para mantener al jugador ocupado, y lo bastante sencilla para que un principiante aprenda a jugar. El juego es divertido, y hay mucho campo para mejoras, con efectos sonoros y animación extra, pero el mayor problema es que sólo ejercita un tipo de habilidad, y puede llegar a resultar aburrido. En la forma en que está escrito, el mayor reto es la dificultad de colocar el balancín debajo de la rana que cae, y es lo que da al juego cierto interés inicial. Una vez que se ha aprendido a controlar esto, se puede hacer muy poco para aumentar el número de insectos comidos, exceptuando el conseguir que la rana no pare de botar. En esta versión, el balancín no sirve de mucho, ya que se utiliza como una simple raqueta. Sin embargo, esto no es difícil de mejorar.

Pensando un momento, veremos que la rana que es catapultada hacia el aire, lo hará en una dirección que depende del tipo de balancín que la impulsa. De momento, la rana sale del balancín con la misma velocidad horizontal de la rana que aterriza. Esto quiere decir que, si una rana que se mueve hacia la izquierda de la pantalla aterriza en el balancín, la

rana catapultada continúa moviéndose hacia la izquierda. No es difícil darse cuenta de que una rana catapultada desde un balancín izquierda debe moverse hacia la derecha, y una rana catapultada desde un balancín derecha debe moverse hacia la izquierda, independientemente de la rana que llega. Hacer este cambio es fácil, añadiendo las siguientes líneas a la subrutina 4500:

```
4530 IF M=1 THEN  LET B=ABS (B):
       GO TO 4550
4540 LET B=-(ABS (B))
```

Así cambiamos la forma de botar en el balancín, y el juego es más interesante, pero no se requiere más destreza que antes para jugarlo. Para que sea así, es obvio que el jugador necesita poder controlar qué tipo de balancín quiere. Si es factible elegir, el jugador puede dirigir la rana hacia el lado en que pueda comer más insectos. Las líneas siguientes se añaden para que el jugador elija por medio de la flecha arriba:

```
3030 IF INKEY$="7" THEN
      GO SUB 4700
4700 REM vuelo
4710 IF M=1 THEN  GO TO 4750
4720 LET S$=L$
4730 LET M=1
4740 GO TO 4790
4750 LET S$=R$
4760 LET M=0
4790 PAUSE 10
4799 RETURN
```

El tiempo de espera de la línea 4790 de la subrutina 4700, se necesita para dar tiempo razonable a la tecla para cambiar el balancín que aparece en pantalla cuando pulsa dicha tecla.

Con esta modificación, el juego cambia, porque el jugador no sólo tiene que dirigir el balancín, también tiene que elegir el adecuado, y comer más insectos. Hay otras formas de usar el balancín para controlar el movimiento de la rana. Por ejemplo, la altura del bote puede depender del punto exacto de aterrizaje, la velocidad horizontal puede depender de cómo se estaba moviendo el balancín, la rana puede perder velocidad cada vez que come una mosca..... pero todo esto corre a cargo de su imaginación.

Versión final—listado completo.

```
  10 REM salto de rana
  30 GO SUB 1000
  40 GO SUB 6000
  50 GO SUB 2000
  60 GO SUB 5000
  70 REM bucle
  90 GO SUB 3000
 100 GO SUB 4000
 110 IF GAMEEND<>1
      THEN  GO TO 70
 120 GO SUB 9000
 130 IF AGAIN THEN  RUN
 199 STOP


1000 REM inicio
1010 DIM F$(2)
1100 DATA "A",3,7,14,28,56,112,
     224,192
1110 DATA "B",192,224,112,56,28,
     14,7,3
1120 DATA "C",24,24,60,60,126,
     126,255,255
1130 DATA "D",36,24,255,255,126,
     60,102,66
1140 DATA "E",130,84,56,186,254,
     130,68,130
1210 FOR I=1 TO 5
1220 READ C$
1230 FOR J=0 TO 7
1240 READ D
1250 POKE USR C$+J,D
1260 NEXT J
1270 NEXT I
1300 DIM L$(70)
1310 DIM R$(70)
1320 LET L$="[    A              ]"
     +"[                     A    ]"
     +"[                          ]"
     +"[      AC]"
1330 LET R$="[ B                  ]"
     +"[                     B    ]"
     +"[                          ]"
     +"[       CB]"
1400 LET M=1
```

```
1410 LET S$=L$
1420 LET X=21
1430 LET FROG=1
1440 LET GAMEEND=0
1450 LET SC=0
1499 RETURN


2000 REM imprimir las moscas
2010 INK 6
2020 PAPER 6
2030 BORDER 6
2040 CLS
2050 LET C=0
2060 FOR V=2 TO 10 STEP 2
2070 LET C=NOT C
2080 FOR U=2 TO 26 STEP 2
2090 PRINT AT V,U-C; INK 1;
     "[D]";
2100 NEXT U
2110 NEXT V
2199 RETURN


3000 REM mover el balancin
3010 IF INKEY$="5" AND X>0 THEN
      LET X=X-1
3020 IF INKEY$="8" AND X<25 THEN
      LET X=X+1
3030 IF INKEY$="7" THEN
      GO SUB 4700
3040 PRINT AT 19,X; INK 2;S$;
3050 LET K=X: LET F$="[ E]"
3060 IF M=0 THEN  LET K=X+3:
       LET F$="[E ]"
3070 INK 4
3080 PRINT AT 20,K;F$;
3099 RETURN


4000 REM mover la rana
4010 PRINT AT G,F; PAPER 8;" ";
4020 LET F=F+B
4030 LET G=G+A
4040 LET A=A+P
```

```
4050 IF F>27 OR F<2 THEN
      LET B=-B
4060 IF G<2 THEN  LET G=2:
       LET A=-A
4070 IF G<11 THEN  GO SUB 7000
4080 PRINT AT G,F; PAPER 8;
     "[E]";
4090 IF G>18 THEN  GO SUB 4500
4099 RETURN


4500 REM balancin
4505 PRINT AT G,F; PAPER 8;" ";
4510 LET D=INT (F)-X-M
4520 IF D<0 OR D>2 THEN
      GO SUB 8000: RETURN
4530 IF M=1 THEN  LET B=ABS (B):
       GO TO 4550
4540 LET B=-(ABS (B))
4550 IF M=1 THEN  GO TO 4590
4560 LET M=1
4570 LET S$=L$
4580 GO TO 4610
4590 LET M=0
4600 LET S$=R$
4610 LET A=-A
4620 LET G=18
4630 LET F=K+1
4640 BEEP .1,-15
4699 RETURN


4700 REM vuelo
4710 IF M=1 THEN  GO TO 4750
4720 LET S$=L$
4730 LET M=1
4740 GO TO 4790
4750 LET S$=R$
4760 LET M=0
4790 PAUSE 10
4799 RETURN

5000 REM arranque
5010 GO SUB 3000
5020 INK 0: PAPER 0
```

```
5030 PRINT AT 15,26;"[      ]":
     REM graficos shift 8
5040 INK 4: PAPER 6
5050 FOR F=30 TO 24 STEP -.1
5060 PRINT AT 14,F;"[E ]";
5070 GO SUB 3000
5080 NEXT F
5090 FOR G=14 TO 18 STEP .5
5100 PRINT AT G-1,24;" ";
5110 PRINT AT G,24;"[E]";
5120 GO SUB 3000
5130 NEXT G
5140 LET F=24
5150 LET G=18
5155 LET C=RND*1
5160 LET A=.4+C/10
5170 LET B=-(.1+C/2.5)
5180 LET P=.008
5190 PRINT AT 0,10;"rana ";FROG;
     " tanteo=";SC
5200 LET A=A+A/DF
5210 LET B=B+B/DF
5220 LET P=P+P/DF
5230 GO SUB 4500
5299 RETURN


6000 REM titulo
6010 PAPER 6
6020 CLS
6030 INK 4
6040 PRINT AT 2,3;
     "S A L T O  ";
     " D E   R A N A"
6050 INK 0
6060 PRINT AT 6,0;
     "En este juego debes ";
     "comerte las"
6070 PRINT "moscas [D]"
6080 PRINT
6090 PRINT "haciendo saltar ";
     "a tus dos"'"ranas [E]"
6100 PRINT "del balancin,";
     " usando las teclas"
6110 PRINT "de cursor ";
     "izquierdo y derecho."
```

```
6111 PRINT "La tecla de cursor";
     " hacia arriba"'"sirve ";
     "para bascular ";
     "el balancin."
6120 PRINT "Tienes diez ranas";
     " para limpiar"
6130 PRINT "la pantalla."
6140 PRINT "Elige el nivel de";
     " dificultad"
6150 INPUT "1 (dificil) A ";
     "10 (facil)";DF
6160 IF DF<1 OR DF>10 THEN
      GO TO 6150
6199 RETURN

7000>REM vuelo de busqueda
7010 IF ATTR (G,F)<>49 THEN
      RETURN
7020 BEEP .1,-20
7030 LET SC=SC+1
7040 PRINT AT 0,10;"rana ";FROG;
     " tanteo=";SC
7050 IF SC=65 THEN
      LET GAMEEND=1
7099 RETURN

8000 REM error
8010 PRINT AT G,F; PAPER 8;" "
8020 LET FROG=FROG+1
8030 IF FROG>10 THEN
      LET GAMEEND=1: RETURN
8040 BEEP .1,0
8050 IF M=0 THEN  LET M=1:
       LET S$=L$
8060 GO SUB 5000
8099 RETURN

9000 REM fin de juego
9010 INK 0
9020 CLS
9030 PRINT AT 10,1;
9040 IF SC<10 THEN
      PRINT "Hasta un renacuajo"
      '" lo haria mejor":
       GO TO 9400
```

```
9050 IF SC<20 THEN
      PRINT "Tenian tres ";
      "patas tus ranas ?":
       GO TO 9400
9060 IF SC<40 THEN
      PRINT "No esta mal para";
      " un sapo miope":
       GO TO 9400
9070 IF SC<50 THEN
      PRINT "Bastante bueno ";
      "considerando....":
       GO TO 9400
9080 IF SC<60 THEN
      PRINT "Una gran operacion"
      ;" de vuelo"'" comiendo":
       GO TO 9400
9090 PRINT "Intentalo metiendo";
     " a las"'" ranas olimpicas"
9400 PRINT AT 15,5;"Tu tanteo ";
     SC
9410 PRINT AT 16,1;"Otro juego";
     "(s/n) ?": INPUT  LINE A$
9420 IF A$(1)<>"n" AND
     A$(1)<>"s" THEN  GO TO 9410
9430 IF A$(1)="s" THEN
      LET AGAIN=1
9440 IF A$(1)="n" THEN
      LET AGAIN=0
9499 RETURN
```

4. Haciendo la rana

Una rana tiene el papel más importante de este capítulo, del mismo modo que en el juego del capítulo anterior. Este juego es una versión de un juego de «marcianitos» muy conocido.

El objeto del mismo es conducir a una rana a través de una calle transitada, esquivando el tráfico que le viene de frente, y después a través de un río con una gran corriente, saltando sobre troncos que flotan en él. La escritura de este juego en BASIC es una buena oportunidad de explicar y experimentar un nuevo tipo de animación: animación por deslizamiento de pantalla (en inglés: *Scrolling animation*).

Animación por deslizamiento de pantalla

La palabra *Scrolling* se ha convertido en algo muy familiar en las pantallas de casi todos los ordenadores. Estamos acostumbrados a que cuando un texto llena toda la pantalla del ordenador, la siguiente sentencia PRINT hará que toda la pantalla suba una línea, de modo que se pierde la primera línea de arriba, y queda libre una línea en la parte de abajo. Lo que ya no es tan conocido es que este enrollamiento o deslizamiento (en inglés Scroll) pueda ser utilizado como un método de animación. Los problemas que se encuentran cuando se intenta imprimir un gran número de caracteres lo bastante rápido para hacer posible el movimiento, ya han aparecido en los dos primeros juegos de este libro. Teniendo en cuenta estos problemas, es fácil ver que el deslizamiento de pantalla es una forma de mover muchos caracteres en poco tiempo, y con poco esfuerzo de programación. Por ejemplo:

```
10 CLS
20 PRINT AT 21,31;
30 POKE 23692,255
40 PRINT TAB RND*31;"*";
50 GO TO 20
```

Este programa imprime asteriscos en posiciones aleatorias en la parte de abajo de la pantalla, y los mueve hacia arriba haciendo uso del deslizamiento. Para prevenir la pregunta del Spectrum: Scroll? cada vez que se ha llenado una pantalla de asteriscos, la línea 30 reestablece las pertinentes variables del sistema, en cada ocasión que se entra en el bucle.

Este programa demuestra que, utilizando este tipo de animación, es

posible mover un gran número de objetos. De hecho, el mayor problema es conseguir ver las cosas. Esto se puede arreglar imprimiendo los objetos al final de una pantalla, y borrándolos antes de la siguiente.

Este tipo de animación es el mejor candidato cuando un juego consiste en muchos objetos moviéndose a igual velocidad en la misma dirección, y puede haber, quizá, uno o dos objetos con un tipo diferente de movimiento. En el caso que nos ocupa, los coches en la calle y los troncos en el río constituyen el mayor número de objetos móviles, y la rana saltarina a través del tráfico y el río es un único objeto que se mueve contrariamente al resto.

El diseño del juego

La elección de esta forma de animación para el juego implica que los coches y los troncos han de moverse verticalmente hacia arriba de la pantalla, y la rana horizontalmente. Para que el juego resulte medianamente entretenido, tendrá que haber, al menos, dos carriles de coches, y dos bandas de troncos flotantes. Saliendo de la parte izquierda de la pantalla, el jugador tiene que conseguir que la rana salte entre el tráfico y sobre la línea segura de separación entre la calle y el río. El río tiene que ser atravesado saltando sobre los troncos. Hay que observar que mientras está sobre un tronco, es llevada por la corriente. La rana debe llegar a la otra orilla antes de que el tronco sobre el que está sentada se deslice fuera del borde superior de la pantalla. Para aumentar la dificultad, y apresurar al jugador a mover la rana, se puede imponer un tiempo límite. Por otra parte, aunque la rana puede avanzar hacia la derecha a traves de la calle y el río, hacia arriba y hacia abajo de la pantalla, no puede moverse hacia la izquierda para ponerse a salvo. Una vez que la rana ha empezado a avanzar, ¡debe continuar!. La disposición del juego se puede ver en la Figura 4.1.

Código máquina

Es evidente que, para que el juego sea bueno, hay que encontrar algún modo de hacer que los dos carriles de tráfico, y las dos líneas de troncos se muevan en proporciones diferentes, y esto sólo se puede hacer empleando código máquina. Necesitamos una subrutina que deslice sólo una columna del texto, sin afectar al resto de la pantalla. Al llamar a esta subrutina, cada carril de coches y cada línea de troncos puede «enrollarse» independientemente del resto.

Quizá la mejor forma de explicar el funcionamiento de esta subrutina en código máquina, sea dando un programa equivalente en BASIC. Se podría pensar que es muy fácil conseguir que sólo se «enrolle» una columna, ya que sólo hay que hacer que cada carácter de la columna suba una línea, pero en la práctica no lo es, debido al modo en que el Spectrum almacena la pantalla en su memoria. La manera más sencilla de resolver el problema de dónde almacenar cada carácter, es computar

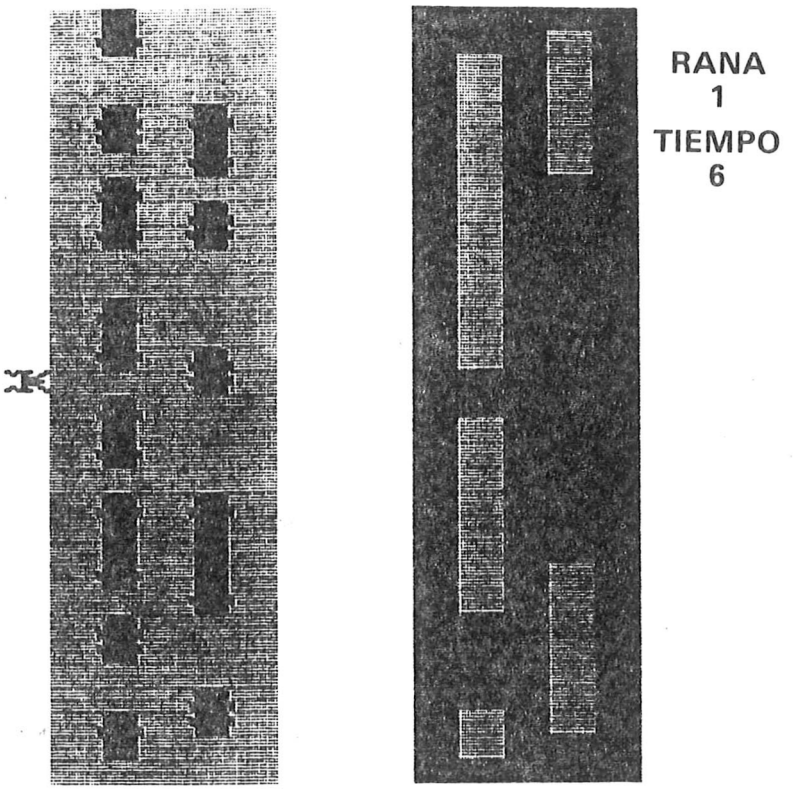


Fig. 4. 1

todas las direcciones que se van a utilizar en la subrutina en código máquina, y guardarlas en una tabla. Esto evita enfrentarse a grandes cálculos dentro de la subrutina y además se consigue mayor rapidez y brevedad. (Si desea conocer más acerca de la pantalla del Spectrum, y cómo se almacena, puede consultar «An Expert Guide to the Spectrum» por Mike James, publicado por Granada).

Al ejecutar el siguiente programa, se puede ver cómo en la columna 10 se desliza un carácter cada vez, muy despacio. La línea 5 reserva espacio para la tabla de direcciones en la parte superior de la memoria. (Si el Spectrum es de 16 K, hay que reducir este valor y el de la línea 6 restando 32767). La variable S almacena la dirección de comienzo de cada línea de pixels que compone las filas del carácter de dicha tabla. La subrutina 8600 es la responsable de deslizar la columna C.

```
50 GO SUB 8600
60 GO TO 40

8500 DEF FN M(L,C,R)=16384+2048*
     INT (L/8)+32*(L-8*INT (L/8))+256*R+C
8505 LET I=0
8510 FOR L=0 TO 21
8520 FOR R=0 TO 7
8530 POKE S+I,FN M(L,0,R)-INT
     (FN M(L,0,R)/256)*256
8535 POKE S+I+L,INT (FN M(L,0,R)/256)
8540 LET I=I+2
8550 NEXT R
8560 NEXT L
8565 POKE I+S,0
8566 POKE I+S+L,0
8570 RETURN

8600 LET I=0
8610 LET A=PEEK (I+16+S)+256*
     PEEK (I+16+S+L)+C
8614 IF A=C THEN GO TO 8650
8615 LET D=PEEK (A)
8620 LET A=PEEK (I+S)+256*PEEK (I+S+L)+C
8625 POKE A,D
8630 LET I=I+2
8640 GO TO 8610
8650 RETURN
```

El final de la tabla está marcado por una dirección con O. La función FN M (L, C, R) devuelve la dirección de la posición de memoria que guarda los ocho puntos que forman la fila R del carácter de la columna C y línea L (esto se describe íntegramente en *The Expert Guide to the Spectrum*).

Aunque este programa funciona, es muy lento, y tal como está no es muy adecuado para el juego. Es claro que la subrutina 8600 ha de ser convertida a código máquina de la siguiente forma:

```
23296 0,0                       NUMERO DE COLUMNA
23298 221,33,200,253            LD IX,64968
23302 42,0,91           LOOP    LD HL,(23296)
23305 221,94,16                 LD E,(IX+16)
23308 221,86,17                 LD D,(IX+17)
23311 62,0                      LD A,0
```

```
23313 186                       CP D
23314 200                       RET Z
23315 25                        ADD HL,DE
23316 126                       LD A,(HL)
23317 221,94,0                  LD E,(IX+0)
23320 221,86,1                  LD D,(IX+1)
23323 42,0,91                   LD HL,(23296)
23326 25                        ADD HL,DE
23327 119                       LD (HL),A
23328 221,35                    INC IX
23330 221,35                    INC IX
23332 195,6,91                  JP LOOP
```

Esta subrutina en código máquina es casi el equivalente exacto de la subrutina 8600. Ha sido escrita para ser almacenada en el «buffer» de la impresora, que es un área de memoria libre muy conveniente para estas subrutinas. La instrucción almacenada en la dirección 23298, carga el registro IX con la dirección del principio de la tabla; si se utiliza un Spectrum de 16 K, debe cambiarse así:

23298 221, 33, 201, 125

Utilizando código máquina, el almacenar el código máquina en el «buffer» de la impresora es simplemente cuestión de añadir sentencias a la subrutina 8500, y luego escribir una nueva versión de la subrutina 8600. Sin embargo, antes de hacer esto es conveniente recalcar que es sólo una solución parcial del problema del deslizamiento de una columna de texto, porque los atributos no se deslizan junto con las figuras. Una subrutina en código máquina que enrolle los atributos de una columna dada, es fácil de escribir, porque el Spectrum almacena los códigos de los atributos en un orden muy simple:

```
23335 6,21                      LD B,21
23337 42,0,91                   LD HL,(23296)
23340 17,0,88                   LD DE,22528
23343 25                        ADD HL,DE
23344 17,32,0                   LD DE,32
23347 229               LOOP2   PUSH HL
23348 25                        ADD HL,DE
23349 126                       LD A,(HL)
23350 225                       POP HL
23351 119                       LD (HL),A
23352 25                        ADD HL,DE
23353 5                         DEC B
23354 194,51,91                 JPNZ LOOP2
23357 201                       RET
```

Esta subrutina se ha escrito para ser guardada en el «buffer» de la impresora a continuación de la subrutina de deslizamiento de caracteres.

Las versiones finales de las subrutinas 8500 y 8600 se dan más abajo junto a un programa principal que sirve como demostración:

```
 5 CLEAR 64967
 6 LET S=64968
10 PRINT AT 21,9;"aaa"
30 GO SUB 8500
40 LET C=10
50 GO SUB 8600
55 PRINT PAPER RND*7; INK RND*7
   AT 21,10;CHR$ (RND*32+32)
60 GOTO 40

8500 DEF FN M(L,C,R)=16384+2048*
     INT (L/8)+(L-8*INT (L/8))*
     32+256*R+C
8505 LET I=0
8510 FOR L=0 TO 21
8520 FOR R=0 TO 7
8530 POKE S+I,FN M(L+1,0,R)-
     INT (FN M(L+1,0,R)/256)*256
8535 POKE S+I+1,
     INT (FN M(L+1,0,R)/256)
8540 LET I=I+2
8550 NEXT R
8560 NEXT L
8565 POKE I+S,0
8566 POKE I+S+1,0
8568 RESTORE 8570
8570 DATA 0,0,221,33,200,253,42,
     0,91
8572 DATA 221,94,16,221,86,17,62
     ,0,186
8574 DATA 200,25,126,221,94,0,
     221,86,1
8576 DATA 42,0,91,25,119,221,35,
     221,35,195,6,91
8578 DATA 6,21,42,0,91,17,0,88,
     25,17,32,0
8579 DATA 229,25,126,225,119,25,
     5,194,51,91,201
8580 FOR I=23296 TO 23357
8585 READ D
8586 POKE I,D
8587 NEXT I
8588 RETURN
```

```
8600 POKE 23296,COL
8610 LET E=USR 23298
8620 LET E=USR 23335
8630 RETURN
```

Como antes, la subrutina 8500 establece una tabla de direcciones, pero ahora también almacena el código máquina para las dos rutinas, en el «buffer» de la impresora. Para hacer que la columna C se deslice, todo lo que hay que hacer es asignar a C el valor correcto, y después GOSUB 8600. De esta forma, la subrutina 8600 enrolla la pantalla a una velocidad razonable. Una vez visto este programa en acción, no es difícil imaginar cómo usarlo para animar el tráfico y las ranas en «Haciendo la rana».

Programa principal y subrutina 7000 —La rutina del título

El programa principal de este juego es un poco diferente de los anteriores, y contiene algunas sentencias que no son simples llamadas a subrutinas

```
 10 REM haciendo la rana
 12 CLEAR 64967
 14 LET S=64968
 15 PRINT AT 10,3;"ESTOY "
    ;"PREPARANDO EL JUEGO"
 16 GO SUB 8500
 20 GO SUB 7000
 30 FOR F=1 TO 3
 40 GO SUB 1000
 50 PAPER 0
 60 PRINT AT 2,20;"RANA";
    AT 3,21;F
 70 GO SUB 400
 80 REM BUCLE
 90 GO SUB 3000
100 GO SUB 6000
110 PAPER 0
115 GO SUB 600
120 IF TM-5>0 THEN  PRINT
      INK 7;AT 5,20;"TIEMPO";
      AT 6,21;INT (TM)-5
130 IF TM-5<=MAX AND
    NOT GAMEEND THEN  GO TO 80
140 GO SUB 6600
145 NEXT F
150 GO SUB 8000
```

```
17Ø IF AGAIN THEN  RUN
199 STOP
```

Las líneas 12 a 16 reservan el espacio para la tabla de valores necesitada por el deslizamiento en código máquina. También establecen S para que sea la dirección del comienzo de la tabla, y llaman a la subrutina 8500, que calcula la tabla. A causa de este cálculo, cuando se ejecuta el programa habrá una pausa con la pantalla en blanco antes de que el título aparezca.

La subrutina 7000 simplemente imprime las instrucciones del juego. El resto del programa principal tiene la forma de dos bucles animados. Las líneas 30 a 145 son un bucle FOR que repite el juego tres veces con tres ranas diferentes. El bucle interior, líneas 80 a 130, es el bucle de animación. La subrutina 1000 llamada por la línea 40 hace el trabajo habitual de inicializar todo lo que necesite ser inicializado. Las líneas 50 y 60 imprimen el número de la rana en juego, antes de que el juego propiamente dicho comience. El bucle de animación llama a la subrutina 3000 para animar la calle y el río, y luego llama a la subrutina 6000 para permitir al jugador mover la rana. Finalmente, la línea 120 imprime el tiempo actual, y la línea 130 comprueba el final del juego. La subrutina 6000 señala el fin de otra rana o el éxito al llegar al otro lado. El juego entero llega a un final con la subrutina 8000, que también pregunta si se quiere volver a jugar.

El único punto importante que queda por mencionar es el empleo de TM—5 en la sentencia IF en las líneas 120 y 130. Esto se debe a que durante los cinco primeros segundos de ejecución del bucle de animación, la rana no puede moverse. Estos cinco primeros segundos son necesarios para dejar que los coches y los troncos suban por la pantalla para que el juego resulte divertido, y desde el principio tenga dificultades. Por tanto, la secuencia del comienzo es: se dibujan la calle y el río, el tráfico y los troncos son animados durante cinco segundos, y por fin aparece la rana, y empieza a contar el tiempo.

La subrutina 8500 ya se ha descrito anteriormente, pero aquí se repite en su forma final:

```
85ØØ DEF FN M(L,C,R)=16384+2Ø48*
     INT (L/8)+(L-8*INT (L/8))*
     32+256*R+C
8505 LET I=Ø
8510 FOR L=Ø TO 21
8520 FOR R=Ø TO 7
8530 POKE S+I,FN M(L+1,Ø,R)-
     INT (FN M(L+1,Ø,R)/256)*256
8535 POKE S+I+1,
     INT (FN M(L+1,Ø,R)/256)
8540 LET I=I+2
8550 NEXT R
```

```
8560 NEXT L
8565 POKE I+S,0
8566 POKE I+S+1,0
8568 RESTORE 8570
8570 DATA 0,0,221,33,200,253,42,
     0,91
8572 DATA 221,94,16,221,86,17,62
     ,0,186
8574 DATA 200,25,126,221,94,0,
     221,86,1
8576 DATA 42,0,91,25,119,221,35,
     221,35,195,6,91
8578 DATA 6,21,42,0,91,17,0,88,
     25,17,32,0
8579 DATA 229,25,126,225,119,25,
     5,194,51,91,201
8580 FOR I=23296 TO 23357
8585 READ D
8586 POKE I,D
8587 NEXT I
8588 RETURN
```

La rutina del título es tan simple, que damos el listado sin comentarios:

```
7000 REM titulo
7010 INK 5
7020 PAPER 0
7030 BORDER 0: CLS
7040 PRINT AT 0,2;"HACIENDO LA";
     " RANA"
7050 PRINT AT 3,1;"En este ";
     "juego"
7060 PRINT TAB 3;"tienes que ";
      "cruzar"
7070 PRINT TAB 1;"tres ranas ";
     "a traves de"
7080 PRINT TAB 2;"una carretera"
     ;" muy transitada"
7090 PRINT TAB 1;"y un ";
     "caudaloso rio"
7095 PRINT : PRINT
7100 PRINT TAB 1;"usa las ";
     "teclas de cursor"
7110 PRINT TAB 3;"para subir "
7120 PRINT TAB 4;"bajar e ir ";
```

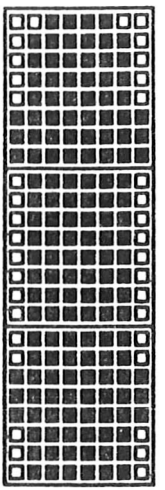
```
      "a la derecha"
7124 PRINT : PRINT
7125 PRINT TAB 1;"No te dejes";
      " atropellar"
7130 PRINT TAB 2;"ni caigas";
      " en el rio"
7145 PRINT : PRINT
7150 PRINT  FLASH 1;AT 21,5;
       "Para comenzar"
7155 PRINT  FLASH 1;AT 21,5;
       "pulsa cualquier tecla"
7166 IF INKEY$="" THEN
       GO TO 7160
7199 RETURN
```

Subrutina 1000 —Sección de inicialización

La subrutina 1000 empieza como siempre definiendo los caracteres gráficos e inicializando las variables, pero en este caso, es también responsable de imprimir las bandas de color que representan la calle y el río:

```
1000 REM inicializacion
1010 DATA "A",60,126,126,126,
     126,255,255,255
1020 DATA "B",126,126,126,126,
     126,126,126,126
1030 DATA "C",126,126,255,255,
     255,126,126,126
1040 DATA "D",185,82,28,30,28,82
     ,185,0
1050 DATA "E",255,255,255,255,
     255,255,255,255
1060 RESTORE 1000
1070 FOR Q=1 TO 5
1080 READ C$
1090 FOR J=0 TO 7
1100 READ D
1107 CLS
1110 POKE USR C$+J,D
1120 NEXT J
1130 NEXT Q
1140 LET A=32: LET B=32
1150 LET C=32: LET D=32
1160 LET X=2: LET Y=15
```

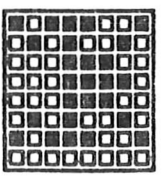
```
1180 FOR I=0 TO 21
1190 PAPER 2: INK 2
1200 PRINT AT I,3;"      ";
1210 PAPER 5: INK 5
1220 PRINT AT I,11;"      ";
1230 NEXT I
1240 LET GAMEEND=0
1250 LET MAX=100
1299 RETURN
```



[A]

[B]

[C]



[D]

[E]

Fig. 4. 2

Caracteres gráficos para el camión, la rana y un tronco.

Las líneas 1010 a 1130 definen los coches, los troncos, y la rana. «[A]» es la parte delantera de un coche o un camión, «[B]» es la parte central del camión, y «[C]» es el final del coche o el camión. Se puede ver cómo se unen estos caracteres en la Figura 4.2. «[D]» es la rana, y «[E]» es un simple bloque sólido que se emplea para hacer los troncos (ver figura 4.2). Las líneas 1140 y 1150 inicializan las variables comprendidas entre A y D con el valor 32 que es el código ASCII del carácter espacio. Estas variables son empleadas por la subrutina 3000 que

se describe en la siguiente sección. X e Y son las coordenadas actuales de la rana, que son inicializadas en la línea 1160.

Las líneas 1180 a 1230 imprimen dos bandas de color a lo largo de la pantalla. La banda roja de la izquierda representa la calle, y la banda azul de la derecha representa el río. Por último, la línea 1240 asigna FALSO (es decir Ø) a GAMEEND, y la línea 1250 establece el tiempo límite del juego en 25 segundos.

Las subrutinas que ponen el tiempo a cero y llevan la cuenta del mismo, son la 400 y 600, respectivamente:

```
400 REM tiempo a cero
410 POKE 23674,0
420 POKE 23673,0
430 POKE 23672,0
440 LET TM=0
499 RETURN


600 REM obtencion del tiempo
610 LET TM=(65536*PEEK 23674+
    256*PEEK 23673+PEEK 23672)/
    50
699 RETURN
```

Subrutina 3000 y subrutinas asociadas

La escritura de este programa se hace en dos partes: generar y enrollar la calle y el río, y mover la rana. Los problemas de mover la rana serán tratados más tarde. La animación de la calle y el río corresponde a la subrutina 3000, que tratamos en esta sección, junto con sus rutinas asociadas.

```
3000 REM movimiento
3030 LET T=A: LET PSTART=.4:
      LET PEND=.5: GO SUB 2000:
       LET A=T
3040 LET Q=A: LET COL=4:
      GO SUB 4000
3050 LET T=B: LET PSTART=.16:
      LET PEND=.5: GO SUB 2000:
       LET B=T
3060 LET Q=B: LET COL=6:
      GO SUB 4000
3070 LET T=B: LET PSTART=.2:
      LET PEND=.5: GO SUB 2000:
```

```
       LET B=T
3080 LET Q=B: LET COL=6:
      GO SUB 4000
3090 LET T=C: LET PSTART=.16:
      LET PEND=.2: GO SUB 5000:
       LET C=T
3100 LET Q=C: LET COL=12:
      GO SUB 4000
3110 LET T=C: LET PSTART=.1:
      LET PEND=.2: GO SUB 5000:
       LET C=T
3120 LET Q=C: LET COL=12:
      GO SUB 4000
3130 LET T=D: LET PSTART=.16:
      LET PEND=.2: GO SUB 5000:
       LET D=T
3140 LET Q=D: LET COL=14:
      GO SUB 4000
3145 LET T=D: LET PSTART=.1:
      LET PEND=.2: GO SUB 5000:
       LET D=T
3150 LET Q=D: LET COL=14:
      GO SUB 4000
3199 RETURN
```

A primera vista, esta subrutina parece complicada, pero es sólo porque el propósito de las subrutinas 2000, 5000, y 4000 no es demasiado evidente. Para entender lo que pasa, hay que pensar qué se requiere en cada etapa de la animación. Consideremos, por ejemplo, el primer carril de la calle. En cada paso de la animación, hemos de decidir entre imprimir un blanco, el principio de un coche o un camión, la parte central de un camión, o el final de un coche o camión. Esto es lo que hace la subrutina 2000 cada vez que es llamada.

PSTART es la posibilidad de que un coche o camión sea generado, es decir, controla el número de coches de la calle. PEND es la probabilidad de que un coche o un camión tenga un final, es decir, controla el número de coches como oposición al de camiones, y la longitud media de los camiones (un coche no es más que un principio de coche seguido de un final de coche, y un camión es un principio de coche, seguido de varias partes centrales de camión, y después un final de coche, ver Figura 4. 2.) T es el código ASCII del último carácter impreso en el carril. Por ejemplo, si el último carácter fuese un espacio (código ASCII = 32), entonces el siguiente carácter puede ser otro espacio, o un principio de coche (claramente no puede ser una parte central de camión ni un final de coche). La decisión de esto es aleatoria, y la probabilidad es determinada por PSTART.

En la subrutina 3000, A se utiliza para guardar el carácter impreso en el carril uno, y la línea 3030 llama a la subrutina 2000, para actualizar dicha variable. La subrutina 2000 devuelve el código del carácter que va a ser impreso. Del mismo modo, B se usa para guardar el carácter del carril dos, y C y D guardan los códigos de los caracteres que se van a imprimir en la primera y segunda columna de troncos, respectivamente. La subrutina 5000 hace lo mismo que la 2000, pero para el río, es decir, devuelve el código del siguiente carácter a imprimir, dando el código del último carácter impreso. La acción de estas subrutinas se verá más claramente cuando luego se describan con mayor detalle.

La subrutina 4000 es la que realmente imprime y desliza. COL controla en cuál de las cuatro columnas se va a imprimir al carácter, y Q es el código ASCII del mismo. Ahora es más sencillo comprender la subrutina 3000. La línea 3030 llama a la subrutina 2000 para generar el carácter a imprimir en el primer carril, después la línea 3040 llama a la subrutina 4000 para imprimirlo en la cuarta columna. Del mismo modo, las líneas 3060 a 3080 generan e imprimen dos carácteres de coche en la columna seis. Como la columna seis se enrolla dos veces cada vez que la columna cuatro se desliza una sola vez, parece que se mueve dos veces más deprisa. Las líneas 3090 a 3120 generan e imprimen dos carácteres de tronco en la columna catorce. Como la columna doce se enrolla dos veces mientras la catorce sólo una, el primer conjunto de troncos se mueve dos veces más rápido que el segundo.

Como ya se ha dicho antes, para entender el funcionamiento de la subrutina 3000 hay que entender cómo funcionan las subrutinas a las que llama. La primera de ellas es la 2000:

```
2000 REM informacion del coche
2010 LET OB=T
2020 IF T=32 AND RND*1<PSTART
      THEN  LET OB=144
2030 IF T=144 THEN  LET OB=145
2040 IF T=146 THEN  LET OB=32
2050 IF OB=145 AND RND*1<PEND
      THEN  LET OB=146
2060 LET T=OB
2099 RETURN
```

La línea 2020 comprueba si el último carácter impreso era un blanco, y si es así, el siguiente carácter tiene que ser otro blanco o la parte delantera de un coche. Si el último carácter era un principio de coche, la línea 2030 hace que el siguiente sea la parte central de un camión. La línea 2040 comprueba si era el final de un coche, en cuyo caso, el siguiente carácter será un blanco. Por fin la línea 2050 cambia partes centrales de camión por finales de coche al azar.

La subrutina 5000 es más simple que la 2000:

```
5000 REM informacion de maderos
5010 IF T=32 AND RND*1<PSTART
     THEN  LET T=148
5020 IF T=148 AND RND*1<PEND
     THEN  LET T=32
5099 RETURN
```

Si el último carácter era espacio, entonces la línea 5010 hará que el siguiente carácter sea un tronco con probabilidad PSTART. La línea 5020 hace lo contrario, y cambia un carácter de tronco por espacio con probabilidad PEND. Resulta claro que PSTART es la probabilidad de acabarlo.

La subrutina 4000 llama a la subrutina 8500 para deslizar la columna, e imprimir el siguiente carácter:

```
4000 REM viento
4010 IF COL>6 OR COL<4 THEN
     GO TO 4040
4020 PAPER 2: INK 5
4030 IF Q=32 THEN  INK 2
4035 GO TO 4070
4040 IF COL>14 THEN  GO TO 4070
4045 PAPER 2: INK 2
4050 IF Q=32 THEN
     INK 5: PAPER 5
4060 IF Q=0 THEN  PAPER 5
4070 IF Q=0 THEN  RETURN
4080 GO SUB 8600
4090 PRINT AT 21,COL;CHR$ (Q);
4099 RETURN
```

Si COL es cuatro, cinco o seis (la rana está en la calle), la línea 4020 establece cyan como color de la tinta, y rojo como color del papel. Si el carácter es un espacio, la tinta también es roja, porque se ha establecido así en la línea 4030, y, por tanto, la función ATTR del Spectrum puede distinguir entre un espacio y un vehículo. El mismo procesó se sigue para el río en las líneas 4040 a 4070. Si Q = 0 no se imprime ningún carácter, y el control pasa a la rutina que llamó. La línea 4080 llama a la subrutina 8600 para enrollar la columna apropiada, y luego la línea 4090 imprime el carácter, que tiene como código el guardado en C, en la parte de abajo de esa columna.

La subrutina 8600 ya se ha descrito antes, pero aquí está listada otra vez en su forma final:

```
8600 POKE 23296,C
8610 LET E=USR 23298
```

```
8620 LET E=USR 23335
8630 RETURN
```

Subrutinas 6000 y 6500

La subrutina 6000 tiene que realizar una tarea muy sencilla, y es muy parecida al resto de las subrutinas que mueven objetos alrededor de la pantalla, sin embargo, tiene que vencer algunas dificultades. En particular, el color del fondo empleado para borrar e imprimir la rana, tiene que cambiar de acuerdo con el lugar en que está, en la calle, en el río, o en ninguno de los dos sitios. Además, el movimiento de la rana depende de si está en una columna deslizante o no. Por ejemplo, si la rana está en la primera columna (es decir, en el primer carril de coches), se deslizará hacia arriba con el tráfico. Para que se quede en el mismo sitio, hay que borrar la rana de su antigua posición , y reimprimirla. Pero si la rana está en la tercera o cuarta columna, en ese caso siempre que esté sentada sobre un tronco se enrollará con el resto de los troncos.

```
6000 REM mover la rana
6005 IF TM<5 THEN  RETURN
6010 INK 4
6015 LET K=0
6016 IF X=4 OR X=6 THEN
      LET K=INT (X/2)-1
6018 PRINT AT Y-K,X; PAPER 8;
     " ";
6020 IF INKEY$="7" AND Y>0
      THEN  LET Y=Y-1
6030 IF INKEY$="6" AND Y<21
      THEN  LET Y=Y+1
6040 IF INKEY$="8" AND X<16
      THEN  LET X=X+2:
        GO SUB 6400
6045 IF X=8 OR X=16 THEN
      LET X=X+2
6046 LET Z=ATTR (Y,X)
6047 IF X<11 AND Z=21 THEN
      LET GAMEEND=1
6048 IF X>11 AND Z=45 THEN
      LET GAMEEND=1
6050 PRINT AT Y,X; INK 4;
      PAPER 8;"[D]";
6055 IF X=12 OR X=14 THEN
      GO SUB 6500
6056 IF X=12 THEN  GO SUB 6500
6060 IF X>14 THEN  LET GAMEEND=1
6099 RETURN
```

La línea 6005 hace que la rana no se mueva, hasta que hayan transcurrido cinco segundos desde el comienzo del juego. La línea 6016 busca dónde se ha de imprimir un espacio para borrar la rana de su antigua posición. Si la rana está dentro de la columna uno (X = 4), o la columna dos (X = 6), se enrollará una o dos líneas respectivamente. Cuando la rana es borrada en la línea 6018, se asigna 8 al color del papel, que imprimirá igual que el color que había en esa posición. Las líneas 6020 a 6040 cambian las coordenadas de la rana dependiendo de la tecla de flecha pulsada. La línea 6045 hace que la rana ejecute automáticamente un doble salto hasta la orilla izquierda del río, o desde la orilla derecha del mismo. Después de esta línea, la nueva posición de la rana es X, Y.

La línea 6046 encuentra el color de la posición en la que va a aterrizar la rana, y lo almacena en Z, utilizando la función ATTR. Esta función devuelve un valor de ocho veces el color del papel más el de la tinta. Si Z es 21 y X 11, es decir INK 5 y PAPER 2, entonces la línea 6047 asigna 1 a GAMEEND. Si X es mayor que 11 y Z es 45, es decir PAPER 5 INK 5, la línea 6048 también asigna 1 a GAMEEND. En cualquier otro caso, la línea 6050 imprime la rana utilizando PAPER 8 para dejar el color del fondo igual que estaba.

La línea 6055 comprueba si la rana está en la columna tres (X = 12) o cuatro (X = 14), y, si es así, la subrutina 6500 es llamada para actualizar la coordenada Y de la rana, como resultado del deslizamiento, y la comprobación de si la rana ha alcanzado la parte superior de la pantalla. La línea 6056 llama a la subrutina 6500 por segunda vez si la rana está en la columna tres (X = 12), para considerar el hecho de que la rana se ha deslizado dos veces. Finalmente, la línea 6060 comprueba si la rana ha llegado a salvo al otro lado del río.

Ya sólo queda por dar el listado de la subrutina 6500:

```
6500 REM ver fin juego
6510 IF Y<0 THEN  LET GAMEEND=1:
       RETURN
6520 LET Y=Y-1
6599 RETURN
```

Subrutinas 6600 a 8000

Las subrutinas finales del juego son muy sencillas. La subrutina 6600 comprueba si la rana ha realizado su viaje con éxito o no, y emite un sonido de acuerdo con el resultado. La subrutina 8000 pregunta al jugador si quiere volver a jugar.

```
6600 REM muerte
6610 IF X=12 OR X=14 THEN
       BEEP .5,-20
```

```
6615 IF X=4 OR X=6 THEN
      BEEP .5,-3Ø
6616 IF X<15 THEN  GO TO 662Ø
6617 FOR I=Ø TO 5Ø STEP 8
6618 BEEP .1,I
6619 NEXT I
662Ø GO SUB 4ØØ
663Ø GO SUB 6ØØ
664Ø IF T<1Ø THEN  GO TO 663Ø
6699 RETURN

8ØØØ REM fin del juego
8Ø1Ø CLS
8Ø2Ø PRINT AT 1Ø,1;"Otra ";
     "partida ?"
8Ø3Ø INPUT  LINE A$
8Ø4Ø IF A$(1)<>"s" AND
     A$(1)<>"n" THEN  GO TO 8Ø1Ø
8Ø5Ø LET AGAIN=Ø
8Ø6Ø IF A$(1)="s" THEN
      LET AGAIN=1
8Ø99 RETURN
```

Evaluación y mejoras

El juego es bastante rápido, resulta divertido, y este tipo de animación produce unas imágenes en la pantalla muy interesantes con poco coste de código máquina extra. Sin embargo, el aspecto más descuidado del juego tal vez sean los efectos sonoros. Esto se arregla fácilmente, añadiendo la subrutina 6400:

```
64ØØ REM sonidos
6410 GO SUB 643Ø
6415 PAUSE 2
642Ø GO SUB 643Ø
6429 RETURN
643Ø FOR I=-5 TO 2 STEP 2
644Ø BEEP .ØØ5,I
645Ø NEXT I
646Ø PAUSE 2
647Ø FOR I=1 TO 3
648Ø BEEP .ØØ5,I
649Ø NEXT I
6499 RETURN
```

Así se produce un sonido con rápidas subidas y bajadas de tono. Las líneas 6410 a 6420 llaman dos veces a la segunda parte de la subrutina y consigue un doble sonido con una pequeña pausa en medio. Las líneas 6430 a 6490 hacen un ruido. Para usar la rutina basta cambiar la línea 6040 por:

```
6040 IF INKEY$="8" AND X<16 THEN
     LET X=X+2:GOSUB 6400
```

Al añadir estos simples efectos sonoros, se aumenta la tensión y el interés del juego, y es útil jugar con y sin ellos para ver la diferencia. Aunque éste es el punto en el que nosotros hemos dejado el juego, no quiere decir que no se pueda extender. En particular se puede introducir un sistema de tanteo, basado en el número de ranas que llegan a salvo a la otra orilla, y el tiempo empleado, y después cambiar la subrutina 8000 para imprimir los comentarios adecuados a estos cambios.

La versión final —Listado completo

```
 10 REM haciendo la rana
 12 CLEAR 64967
 14 LET S=64968
 15 PRINT AT 10,3;"ESTOY ";
    "PREPARANDO EL JUEGO"
 16 GO SUB 8500
 20 GO SUB 7000
 30 FOR F=1 TO 3
 40 GO SUB 1000
 50 PAPER 0
 60 PRINT AT 2,20;"RANA";
    AT 3,21;F
 70 GO SUB 400
 80 REM BUCLE
 90 GO SUB 3000
100 GO SUB 6000
110 PAPER 0
115 GO SUB 600
120 IF TM-5>0 THEN
     PRINT  INK 7;
      AT 5,20;"TIEMPO";
      AT 6,21;INT (TM)-5
130 IF TM-5<=MAX AND
    NOT GAMEEND THEN  GO TO 80
140 GO SUB 6600
145 NEXT F
150 GO SUB 8000
```

```
170 IF AGAIN THEN  RUN
199 STOP


400 REM tiempo a cero
410 POKE 23674,0
420 POKE 23673,0
430 POKE 23672,0
440 LET TM=0
499 RETURN


600 REM obtencion del tiempo
610 LET TM=(65536*PEEK 23674+
    256*PEEK 23673+PEEK 23672)/
    50
699 RETURN

1000 REM inicializacion
1010 DATA "A",60,126,126,126,
     126,255,255,255
1020 DATA "B",126,126,126,126,
     126,126,126,126
1030 DATA "C",126,126,255,255,
     255;126,126,126
1040 DATA "D",185,82,28,30,28,82
     ,185,0
1050 DATA "E",255,255,255,255,
     255,255,255,255
1060 RESTORE 1000
1070 FOR Q=1 TO 5
1080 READ C$
1090 FOR J=0 TO 7
1100 READ D
1107 CLS
1110 POKE USR C$+J,D
1120 NEXT J
1130 NEXT Q
1140 LET A=32: LET B=32
1150 LET C=32: LET D=32
1160 LET X=2: LET Y=15
1180 FOR I=0 TO 21
1190 PAPER 2: INK 2
1200 PRINT AT I,3;"      ";
1210 PAPER 5: INK 5
```

```
1220 PRINT AT I,11;"      ";
1230 NEXT I
1240 LET GAMEEND=0
1250 LET MAX=100
1299 RETURN


2000 REM informacion del coche
2010 LET OB=T
2020 IF T=32 AND RND*1<PSTART
      THEN  LET OB=144
2030 IF T=144 THEN  LET OB=145
2040 IF T=146 THEN  LET OB=32
2050 IF OB=145 AND RND*1<PEND
      THEN  LET OB=146
2060 LET T=OB
2099 RETURN


3000 REM movimiento
3030 LET T=A: LET PSTART=.4:
      LET PEND=.5: GO SUB 2000:
       LET A=T
3040 LET Q=A: LET COL=4:
      GO SUB 4000
3050 LET T=B: LET PSTART=.16:
      LET PEND=.5: GO SUB 2000:
       LET B=T
3060 LET Q=B: LET COL=6:
      GO SUB 4000
3070 LET T=B: LET PSTART=.2:
      LET PEND=.5: GO SUB 2000:
       LET B=T
3080 LET Q=B: LET COL=6:
      GO SUB 4000
3090 LET T=C: LET PSTART=.16:
      LET PEND=.2: GO SUB 5000:
       LET C=T
3100 LET Q=C: LET COL=12:
      GO SUB 4000
3110 LET T=C: LET PSTART=.1:
      LET PEND=.2: GO SUB 5000:
       LET C=T
3120 LET Q=C: LET COL=12:
      GO SUB 4000
```

```
3130 LET T=D: LET PSTART=.16:
      LET PEND=.2: GO SUB 5000:
       LET D=T
3140 LET Q=D: LET COL=14:
      GO SUB 4000
3145 LET T=D: LET PSTART=.1:
      LET PEND=.2: GO SUB 5000:
       LET D=T
3150 LET Q=D: LET COL=14:
      GO SUB 4000
3199 RETURN


4000 REM viento
4010 IF COL>6 OR COL<4 THEN
      GO TO 4040
4020 PAPER 2: INK 5
4030 IF Q=32 THEN  INK 2
4035 GO TO 4070
4040 IF COL>14 THEN  GO TO 4070
4045 PAPER 2: INK 2
4050 IF Q=32 THEN  INK 5: PAPER
5
4060 IF Q=0 THEN  PAPER 5
4070 IF Q=0 THEN  RETURN
4080 GO SUB 8600
4090 PRINT AT 21,COL;CHR$ (Q);
4099 RETURN


5000 REM informacion de maderos
5010 IF T=32 AND RND*1<PSTART
      THEN  LET T=148
5020 IF T=148 AND RND*1<PEND
      THEN  LET T=32
5099 RETURN

6000 REM mover la rana
6005 IF TM<5 THEN  RETURN
6010 INK 4
6015 LET K=0
6016 IF X=4 OR X=6 THEN
      LET K=INT (X/2)-1
6018 PRINT AT Y-K,X; PAPER 8;
     " ";
```

```
6020 IF INKEY$="7" AND Y>0
      THEN  LET Y=Y-1
6030 IF INKEY$="6" AND Y<21
      THEN  LET Y=Y+1
6040 IF INKEY$="8" AND X<16
      THEN  LET X=X+2: GO SUB 64
00
6045 IF X=8 OR X=16 THEN
      LET X=X+2
6046 LET Z=ATTR (Y,X)
6047 IF X<11 AND Z=21 THEN
      LET GAMEEND=1
6048 IF X>11 AND Z=45 THEN
      LET GAMEEND=1
6050 PRINT AT Y,X; INK 4;
      PAPER 8;" ";
6055 IF X=12 OR X=14 THEN
      GO SUB 6500
6056 IF X=12 THEN  GO SUB 6500
6060 IF X>14 THEN  LET GAMEEND=1
6099 RETURN


6400 REM sonidos
6410 GO SUB 6430
6415 PAUSE 2
6420 GO SUB 6430
6429 RETURN
6430 FOR I=-5 TO 2 STEP 2
6440 BEEP .005,I
6450 NEXT I
6460 PAUSE 2
6470 FOR I=1 TO 3
6480 BEEP .005,I
6490 NEXT I
6499 RETURN


6500 REM ver fin juego
6510 IF Y<0 THEN  LET GAMEEND=1:
       RETURN
6520 LET Y=Y-1
6599 RETURN

6600 REM muerte
```

```
6610 IF X=12 OR X=14 THEN
      BEEP .5,-20
6615 IF X=4 OR X=6 THEN
      BEEP .5,-30
6616 IF X<15 THEN  GO TO 6620
6617 FOR I=0 TO 50 STEP 8
6618 BEEP .1,I
6619 NEXT I
6620 GO SUB 400
6630 GO SUB 600
6640 IF T<10 THEN  GO TO 6630
6699 RETURN

7000 REM titulo
7010 INK 5
7020 PAPER 0
7030 BORDER 0: CLS
7040 PRINT AT 0,2;"HACIENDO LA";
     " RANA"
7050 PRINT AT 3,1;"En este ";
     "juego"
7060 PRINT TAB 3;"tienes que ";
      "cruzar"
7070 PRINT TAB 1;"tres ranas ";
     "a traves de"
7080 PRINT TAB 2;"una carretera"
     ;" muy transitada"
7090 PRINT TAB 1;"y un ";
     "caudaloso rio"
7095 PRINT : PRINT
7100 PRINT TAB 1;"usa las ";
     "teclas de cursor"
7110 PRINT TAB 3;"para subir "
7120 PRINT TAB 4;"bajar e ir ";
     "a la derecha"
7124 PRINT : PRINT
7125 PRINT TAB 1;"No te dejes";
     " atropellar"
7130 PRINT TAB 2;"ni caigas";
     " en el rio"
7145 PRINT : PRINT
7150 PRINT  FLASH 1;AT 21,5;
      "Para comenzar"
7155 PRINT  FLASH 1;AT 21,5;
      "pulsa cualquier tecla"
```

```
7166 IF INKEY$="" THEN
      GO TO 7160
7199 RETURN

8000 REM fin del juego
8010 CLS
8020 PRINT AT 10,1;"Otra ";
     "partida ?"
8030 INPUT  LINE A$
8040 IF A$(1)<>"s" AND
     A$(1)<>"n" THEN  GO TO 8010
8050 LET AGAIN=0
8060 IF A$(1)="s" THEN
      LET AGAIN=1
8099 RETURN


8500 DEF FN M(L,C,R)=16384+2048*
     INT (L/8)+(L-8*INT (L/8))*
     32+256*R+C
8505 LET I=0
8510 FOR L=0 TO 21
8520 FOR R=0 TO 7
8530 POKE S+I,FN M(L+1,0,R)-
     INT (FN M(L+1,0,R)/256)*256
8535 POKE S+I+1,
     INT (FN M(L+1,0,R)/256)
8540 LET I=I+2
8550 NEXT R
8560 NEXT L
8565 POKE I+S,0
8566 POKE I+S+1,0
8568 RESTORE 8570
8570 DATA 0,0,221,33,200,253,42,
     0,91
8572 DATA 221,94,16,221,86,17,62
     ,0,186
8574 DATA 200,25,126,221,94,0,
     221,86,1
8576 DATA 42,0,91,25,119,221,35,
     221,35,195,6,91
8578 DATA 6,21,42,0,91,17,0,88,
     25,17,32,0
8579 DATA 229,25,126,225,119,25,
     5,194,51,91,201
```

```
8580 FOR I=23296 TO 23357
8585 READ D
8586 POKE I,D
8587 NEXT I
8588 RETURN


8600 POKE 23296,COL
8610 LET E=USR 23298
8620 LET E=USR 23335
8630 RETURN
```

5. La Serpiente

Animar un objeto largo, es normalmente un problema bastante difícil, independientemente del lenguaje que se utilice. Si elegimos BASIC, entonces el problema recae en el intento de cambiar un gran número de pixels, lo suficientemente rápido como para dar la impresión de un movimiento suave. En cambio, si se utiliza ENSAMBLADOR, el principal problema es escribir complicados cálculos que van controlando en qué lugar está cada cosa. Dicho más sencillamente, el problema de intentar mover un objeto largo, es que generalmente implica imprimir, borrar, y reimprimir demasiados carácteres, cada vez que entramos en el bucle de animación.

De todos modos, hay un objeto muy largo, que se puede animar fácilmente, con bastante velocidad, y que además tiene un movimiento complicado y fascinante: La Serpiente. Una serpiente se puede menear a lo largo de la pantalla, de tal modo que la velocidad no depende de su longitud. Si controlamos la dirección del movimiento de la serpiente con las cuatro flechas, sólo tendremos que conducirla por la pantalla. Nos quedan por determinar unas pocas reglas simples, como no permitir a la serpiente que cruce sobre su propia cola, y el programa no es más que un reto al ingenio. En este capítulo se describe el método fundamental de animación de serpientes, y se desarrolla uno de los juegos más entretenidos del libro.

Animación de serpientes

Una serpiente está compuesta de varios carácteres gráficos impresos en línea. Uno de los extremos será la cabeza, y el otro la cola. La dirección del movimiento de la serpiente está gobernada por la dirección del movimiento de la cabeza, y el resto de los carácteres no hará más que seguir a la misma. Por ejemplo, el segundo carácter la sigue un movimiento detrás. En otras palabras, cuando la cabeza se mueve hasta una posición, el segundo carácter se mueve hasta la antigua posición de la cabeza, y del mismo modo, el tercer carácter se mueve hasta la antigua posición del segundo, y así hasta llegar a la cola.

Esta descripción de cómo se mueve una serpiente es bastante exacta, pero da la sensación de que todos los carácteres han de moverse cada vez que se mueve la cabeza. Si éste fuera el caso, el animar una serpiente de cualquier tamaño, sería un problema para código máquina. Es decir, que si todos los carácteres son iguales, basta simplemente con mover la

cabeza y la cola para dar la impresión de que la serpiente entera se mueve. La razón de esto es clara, la cabeza tiene que moverse porque llega a una posición que estaba en blanco (en general, una posición que no formaba parte de la serpiente). Por otro lado, el segundo carácter llega a la posición que antes ocupaba la cabeza, pero si son iguales no hay un cambio visible. Así que, si decidimos no borrar la antigua posición de la cabeza, no hay necesidad de imprimir el segundo carácter en su nueva posición. Este argumento se puede repetir para toda la serpiente hasta llegar a la cola. Para la cola también es válido, pero la antigua posición de la misma ha de ser apagada, ya que no queda ningún carácter de la serpiente para ocuparla.

Resumiendo, si la serpiente está compuesta de carácteres idénticos, podemos moverla sin más que imprimir la nueva posición de la cabeza, y borrar la antigua posición de la cola.

Veamos una demostración simple de este método, siguiendo el programa que anima una serpiente corta, compuesta de letras «S», a lo largo de la pantalla, en una dirección fija:

```
 10 CLS
 20 LET XH=10: LET YH=10
 30 LET XT=0: LET YT=10
 40 PRINT AT YH,XH;"S";
 50 PRINT AT YT,XT;" ";
 60 LET XH=XH+1
 70 LET XT=XT+1
 80 IF  XH>30 THEN  LET XH=1
 90 IF  XT>30 THEN  LET XT=1
100 IF INKEY$<>"" THEN  STOP
110 GO TO 40
```

La línea 20 define la posición de la cabeza en XH, YH, y la línea 30 establece la antigua posición, y la línea 40 imprime la cabeza en su nueva posición, y la línea 50 borra la cola. Las líneas 60 y 70 actualizan las posiciones de la cabeza y la cola, y las líneas 80 y 90 comprueban si las nuevas posiciones siguen dentro de los límites de la pantalla. La línea 100 es una pequeña pausa, y después se repite el bucle. Este ejemplo es muy simple, porque la serpiente se mueve en línea recta, y la actualización es muy sencilla.

En la práctica, los carácteres de la cabeza y la cola son diferentes del resto, pero no da mucho trabajo extra. Si la cabeza es diferente del resto del cuerpo, su antigua posición debe cambiarse por un carácter de cuerpo, y, si la cola es diferente, además de borrar su antigua posición, hemos de reimprimir la nueva. Por tanto, incluso una serpiente «bonita», con cabeza y cola diferenciadas, sólo hace necesaria la impresión de cuatro carácteres, independientemente de su tamaño.

La serpiente dirigida —«Colas»

Aunque mover una serpiente sólo requiere la impresión de cuatro caracteres cada vez que entramos en el bucle de animación, nos queda todavía el problema de llevar la cuenta de todas las posiciones de todos los caracteres de la serpiente. Como el movimiento sólo implica a la cabeza y la cola, podríamos pensar que esto no es necesario. La razón de esta necesidad es que hemos de conocer dónde estaba impresa la cola en cada movimiento. Cuando la serpiente se mueve en línea recta, como en el ejemplo, es fácil conocer la nueva posición de la cola, pero si la dirección de movimiento cambia, es bastante diferente. Cada vez que la serpiente se mueve, la cola se mueve a la posición ocupada por el penúltimo carácter, y, por tanto, hay que conocer su posición. Esto se repite cada vez que la serpiente se mueve, ya que entonces el antepenúltimo carácter se convierte en el penúltimo. En otras palabras, si todas las coordenadas del I—simo carácter están almacenadas en X(I) e Y(I), en cada movimiento las coordenadas se actualizan del modo siguiente:

```
LET XT=X(1)
LET YT=Y(1)
FOR I=N TO 2 STEP -1
LET X(I-1)=X(I)
LET Y(I-1)=Y(I)
NEXT I
```

y

```
LET X(N)=XH
LET Y(N)=YH
```

(donde la serpiente tiene una longitud de N caracteres, las coordenadas de la cabeza son X(N), Y(N), las coordenadas de la antigua posición de la cola son XT YT, y las nuevas coordenadas de la cabeza son XH YH). Al examinar este bucle FOR, vemos que todas las coordenadas de la lista se mueven un lugar, las del primer carácter pasan a ser las del segundo, y así hasta el final. Siguiendo este esquema, la animación de la serpiente se consigue imprimiendo la cabeza en su nueva posición:

```
PRINT AT X(N),Y(N);H$;
```

cambiando el carácter de la antigua posición de la cabeza, por un carácter de cuerpo:

```
PRINT AT X(N-1),Y(N-1);S$;
```

y finalmente, borrando la cola e imprimiéndola en su nueva posición:

```
PRINT AT XT,YT;" ";
PRINT AT X(1),Y(1);T$;
```

(donde H$, S$, y T$ son cadenas que contienen los caracteres de la cabeza, el cuerpo, y la cola, respectivamente)

El problema de este método es que, cada vez que la serpiente se mueve, hemos de cambiar los contenidos de las matrices X(I) e Y(I). Esto es demasiado para un programa escrito en BASIC, y, lo que es peor, la cantidad de trabajo aumenta con la longiud de la serpiente. Utilizando esta idea, y además en BASIC, la serpiente se moverá despacio, casi parecerá paralizada, e incluso dará la impresión de aumentar de tamaño. La solución se puede encontrar utilizando una avanzada «estructura de datos», conocida como «cola». Si está interesado en conocer algo más acerca de esta teoría, puede consultar «The Complete Programmer» , de Mike James (Granada 1983). La idea básica es evitar mover los datos de las matrices X Y, utilizando un par de punteros, uno para las coordenadas de la cabeza, y otro para las coordenadas de la cola. Por ejemplo, si las coordenadas de cada carácter se van a almacenar de nuevo en las listas X Y, siendo Q el índice de los elementos que guardan la cabeza, y Z el índice de los elementos de la matriz que guardan la cola, el procedimiento de actualización resulta:

```
LET Q=Q+1
LET X(Q)=XH
LET Y(Q)=YH

LET Z=Z+1
```

Es decir, que la nueva posición de la cabeza se ha almacenado en un elemento superior al de la antigua posición. Así, cada vez que la cabeza se mueve deja tras de sí un rastro de las coordenadas de sus antiguas posiciones. Por ejemplo, si la cabeza se encuentra actualmente en X(Q-1), Y(Q-1), antes estaba en X(Q-2), Y(Q-2)... y así hasta el final. Este rastro de las coordenadas se puede utilizar para hacer que la cola siga a la cabeza a través de la pantalla, simplemente cambiándola de X(Z), Y(Z) a X(Z + 1), Y(Z + 1) cada vez dentro del bucle. Técnicamente, el rastro de coordenadas y las variables Q y Z, se denominan una «cola». Este término parece apropiado, si pensamos que las coordenadas de la cabeza están pegadas al final de una cola de coordenadas, y las coordenadas de la nueva posición de la cola forman el frente de la «cola». Como la serpiente se mueve hacia delante, las coordenadas bajan por la «cola», para alcanzar la posición actual de la cola.(Fig. 5. 1)

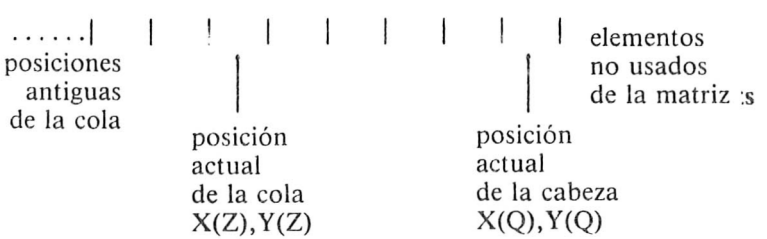


Fig. 5.1

Este método de almacenar las coordenadas de la cabeza presenta un gran problema: como la serpiente se mueve a lo largo de la pantalla, la matriz cada vez ocupa más y más espacio, sin embargo, cada vez, sólo se necesitan los elementos que están comprendidos entre X(Z), Y(Z) y X(Q), Y(Q). El resto de los mismos o está sin usar o guarda antiguas posiciones que no queremos para nada. La solución es hacer una lista lo bastante grande como para guardar todas las cordenadas, de la mayor serpiente que queramos animar, y si alguno de los punteros Z ó Q llega al límite de la matriz, volvemos a ponerlo en 1. El mejor modo de imaginar esto, es pensar que la matriz es circular, y su último elemento es el siguiente al primero. Las posiciones actuales de la cabeza y cola, se mueven alrededor de un círculo, con todas las posiciones de interés almacenadas entre ellas. Gracias a esto, ya tenemos todas las ideas necesarias para escribir juegos basados en serpientes animadas.

Diseño del juego

Hay muchas formas diferentes de utilizar las serpientes animadas como parte del juego. En el juego desarrollado en este capítulo, la simplicidad es el principal objetivo. La idea del juego es la siguiente: el jugador debe guiar a una serpiente que se mueve continuamente por la pantalla, con el fin de comer el mayor número posible de ranas. Cada vez que la serpiente se come una rana, crece un carácter, y así es más difícil y más divertido controlarla. Para aumentar la dificultad, la serpiente no sólo debe evitar cruzar sobre sí misma, debe evitar también comer los sapos venenosos que se esconden entre las ranas. La Figura 5.2 muestra una pantalla típica de este juego. Otros detalles del programa se irán introduciendo según vayamos escribiéndolo.

El Programa Principal

El programa principal de este juego, tiene la forma usual:

```
 10 REM serpiente
 20 CLS
 30 LET AGAIN=0: REM falso
 40 GO SUB 1000
 50 GO SUB 8000
 60 GO SUB 2000
 70 REM bucle
 80 GO SUB 3000
 90 IF NOFOOD OR NOLIVES THEN
     GO TO 110
100 GO TO 70
110 GO SUB 7000
120 IF AGAIN THEN  GO TO 40
130 STOP
```

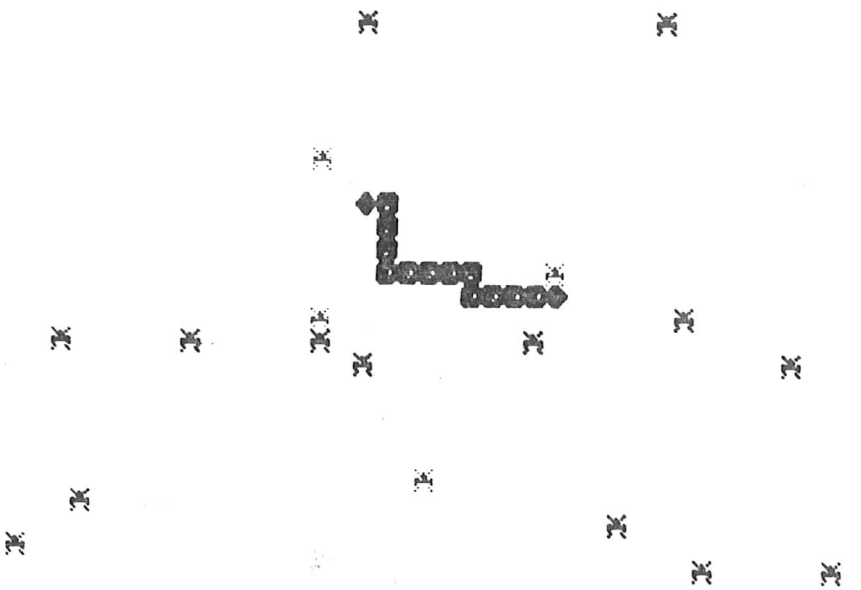


Fig.5. 2

La línea 30 establece FALSO en la variable AGAIN (es decir 0). Esta variable nos indica si el programa se ha ejecutado por primera vez o no. La línea 120 comprueba AGAIN y vuelve atrás si queremos seguir jugando. La línea 40 llama a la subrutina 1000 para establecer los gráficos de usuario, etc... La línea 50 llama a la subrutina 8000 para enmarcar el título en la pantalla, y la línea 60 llama a la subrutina 2000 para dibujar la serpiente, las ranas y los sapos en su posición de comienzo. Las líneas 70 a 1000 forman el bucle de animación. La subrutina 3000 llamada en la línea 80, es la responsable de mover la serpiente, y comprobar si ésta ha comido algo. La línea 90 controla el final del juego. La variable NOFOOD es verdad, si la serpiente se ha comido todas las ranas sin gastar ninguna de sus cinco vidas, y la variable NOLIVES es verdad, si la serpiente ha cruzado sobre sí misma, o ha comido un sapo venenoso cinco veces, y ha perdido sus vidas. Finalmente, la subrutina 7000 termina el juego con un mensaje, y si es apropiado permite al jugador guardar su nombre en una tabla de «mejores puntuaciones».

La subrutina 1000

Los gráficos de usuario que componen la serpiente son algo más complicados de lo que cabía esperar. Tenemos cuatro cabezas diferentes

(gráficos A, D, G, e I, uno para cada dirección de movimiento), también hay cuatro caracteres diferentes para el cuerpo B, E, H y J, en modo gráfico. (fig 5.3). Al seleccionar el cuerpo que apunta aen la misma dirección que la cabeza, e imprimir la cabeza, la serpiente parecerá más llamativa, y llena de movimiento. Sólo se emplea un caracteres para la cola, gráfico C, porque la forma del diamante es la misma en cualquier dirección. El carácter de la rana está definido en la F, y sirve tanto para las ranas como para los saposm venenosos. Las ranas se imprimen en verde, y los sapos en rojo brillante.

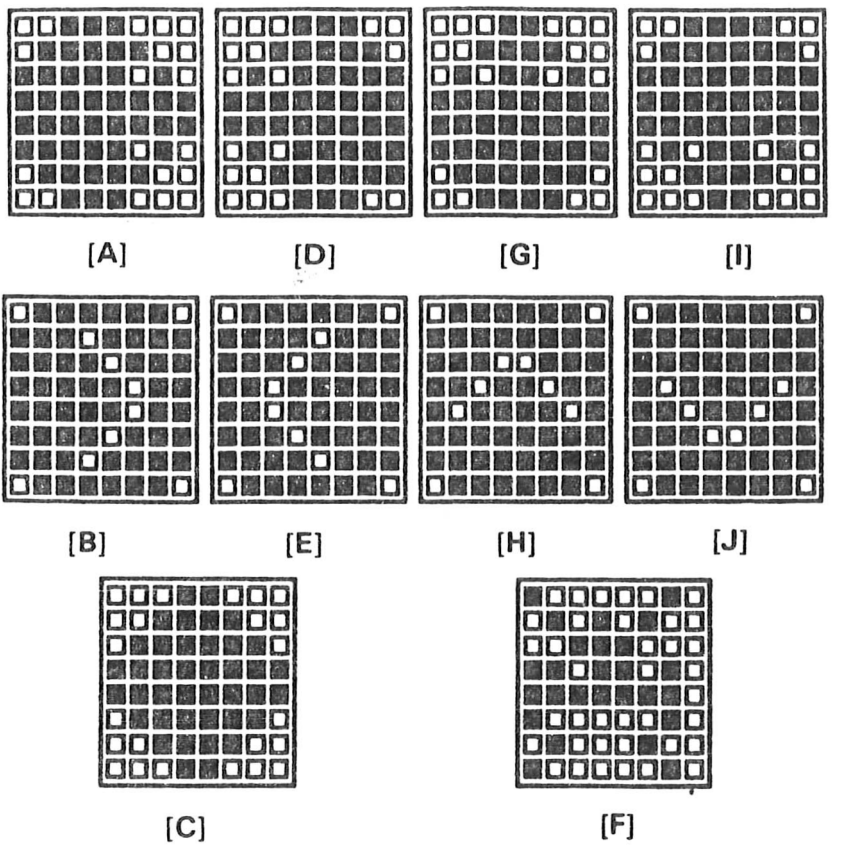


Fig. 5.3.

Caracteres gráficos de: cabeza de la serpiente, cuerpo de la serpiente, cola de la serpiente, y rana.

```
1000 REM inicializacion
1005 RESTORE 1000
1010 DATA "A",56,124,250,255,
     255,250,124,56
1020 DATA "B",126,239,247,251,
     251,247,239,126
```

```
1030 DATA "C",24,60,126,255,
     255,126,60,24
1040 DATA "D",28,62,95,255,
     255,95,62,28
1050 DATA "E",126,247,239,223,
     223,239,247,126
1060 DATA "F",130,84,56,60,
     254,130,68,130
1070 DATA "G",24,60,90,255,
     255,255,126,60
1080 DATA "H",60,255,231,219,
     189,255,255,126
1090 DATA "I",60,126,255,255,
     255,90,60,24
1100 DATA "J",126,255,255,189,
     219,231,255,126
1110 DIM C$(1)
1120 FOR I=1 TO 10
1130 READ C$
1140 FOR J=0 TO 7
1145 READ D
1150 POKE USR C$+J,D
1160 NEXT J
1170 NEXT I
1190 IF AGAIN THEN  GO TO 1260
1200 DIM X(50): DIM Y(50)
1205 LET MAX=50
1210 DIM T$(5,15)
1220 DIM T(5)
1230 FOR I=1 TO 5
1240 LET T(I)=0
1250 NEXT I
1260 LET NOFOOD=0
1270 LET NOLIVES=0
1280 LET HIT=0
1290 LET C=0: LET STOTAL=0
1300 LET SC=0: LET SN=0:
      LET FOOD=0
1399 RETURN
```

La primera parte de esta subrutina de comienzo, establece los gráficos de usuario (líneas 1010 a 1170). La línea 1190 salta la sección de la subrutina que define las matrices, etc..., si no es la primera vez que la subrutina ha sido llamada. Esto es debido a que en el BASIC del Spectrum, si definimos dos veces la misma matriz, nos da error. También las definiciones de los caracteres se podrían saltar, una vez que se

ha utilizado la subrutina, pero la pérdida de tiempo por no hacerlo es mínima. La línea 1200 define las dos matrices X e Y que más tarde emplearemos para almacenar las coordenadas de la serpiente. MAX (línea 1205) se utiliza para fijar la máxima longitud de las matrices, que es también la longitud de la serpiente. T$ y T en las líneas 1210 y 1220 se usarán para guardar la liga de las mejores puntuaciones.

Finalmente, la última parte de la subrutina inicializa algunas variables generales que más tarde se utilizarán. Las líneas 1260 y 1270 asignan FALSO (es decir 0) a los indicadores del final del juego. La variable HIT es VERDADERO (1) si la serpiente no ha chocado contra nada, y FALSO (0), en caso contrario. Las variables C y STOTAL van guardando el tanteo del juego. Cada jugador tiene que comer cinco pantallas de ranas, y dispone de cinco vidas para hacerlo. La variable C lleva la cuenta de las ranas comidas en la pantalla actual, y STOTAL cuenta el número total de ranas comidas en todas las pantallas anteriores. La variable SC es el número de vidas que van quedando. La variable FOOD controla el número de ranas no comidas.

Encuadre del Título

La subrutina 8000 es muy simple, excepto por el uso que hace de los caracteres de la serpiente para escribir la palabra «SERPIENTE» en grandes letras.

```
8000 REM titulo
8005 INK 0: PAPER 6
8006 CLS
8007 BORDER 6
8010 GO SUB 8500
8020 PRINT AT 9,1;
8030 PRINT "Debes guiar tu ";
     "serpiente usando"
8040 PRINT "las cuatro teclas ";
     "de cursor, y"
8050 PRINT "comerte todas las ";
     "ranas verdes ";
8060 PRINT  INK 4;"[F]"
8070 PRINT
8080 PRINT "Si tratas de ";
     "comerte un sapo ";
8090 PRINT  BRIGHT 1; INK 2;
      "[F]"
8100 PRINT " o a ti misma"
8110 PRINT "entonces perderas ";
     "una de tus"
8120 PRINT "cinco vidas"
```

```
8130 PRINT
8140 PRINT "has de comerte ";
     "cinco pantallas"
8150 PRINT "antes de ";
     "convertirte"
8160 PRINT "en una  ";
     "SUPER SERPIENTE"
8170 GO SUB 2200
8180 GO SUB 2300
8190 IF TIME<=50 AND INKEY$=""
      THEN  GO TO 8180
8199 RETURN
```

Las líneas 8060 y 8090 imprimen una rana y un sapo para mostrarlos al jugador. La subrutina 8500 escribe la palabra «serpiente»:

```
8500 REM serpiente
8505 PRINT : PRINT
8510 DATA 8,2,2,1,0,8,2,2,2,0,8,
     2,2,2,0
8520 DATA 3,0,8,1,0,8,2,2,1
8530 DATA 8,0,0,0,0,8,0,0,10,0,
     8,0,0,10
8540 DATA 0,10,8,2,0,0,8,0,0,0
8550 DATA 5,5,5,8,0,8,0,0,10,0,
     8,5,5
8560 DATA 10,0,10,8,0,0,0,5,5,8,
     0
8570 DATA 0,0,0,8,0,8,0,0,10,0,
     8,10,2,2
8580 DATA 0,10,5,8,0,0,8,2,2,0
8583 DATA 0,0,0,8,0,8,0,0,10,0,
     8,0,0
8586 DATA 10,0,10,0,5,8,0,8,0,0,
     0
8590 DATA 3,2,2,2,0,3,0,0,9,0,3,
     0,0,9
8595 DATA 0,9,0,0,3,0,5,5,5,3
8600 RESTORE 8500
8610 FOR I=1 TO 6
8620 PRINT TAB 2;
8630 FOR J=1 TO 24
8640 READ S
8650 IF S=0 THEN  PRINT " ";:
       GO TO 8670
8660 PRINT CHR$ (143+S);
```

```
8670 NEXT J
8680 PRINT
8690 NEXT I
8699 RETURN
```

Las sentencias DATA de las líneas 8500 a 8595 son simples listas que indican los caracteres que se han de imprimir. Para reducir el número de dígitos a teclear en cada sentencia DATA, se ha añadido 143 en cada número para dar el código del carácter. Por ejemplo, un 1 en la sentencia DATA hará imprimir CHR$(144), con la única excepción del 0 que imprime un espacio en blanco. Las sentencias DATA están organizadas en pares, de modo que cada par define una única fila de 24 caracteres. Seis filas de estos caracteres de tamaño estándar se utilizan para formar la palabra «serpiente», en grandes letras. Existen otros métodos para conseguir letras de gran tamaño, pero éste es el más conveniente.

Subrutina 2000 (dibujar escena), 2600 (dibujar serpiente), y 2500 (dibujar comida)

Aunque podría parecer poco claro en este punto del programa, la subrutina 2000 es llamada por otras subrutinas dentro del programa para re-dibujar la escena después de que la serpiente ha intentado comer un sapo venenoso o a sí misma. En este sentido, no es sólo parte de la fase de inicialización del programa, es también parte del bucle de animación. La única modifcación necesaria para que se pueda emplear en ambos casos, es que debe dibujar tantas ranas como nos indique la variable FOOD, pero si ésta es cero, debe serle asignado un nuevo valor al azar, y así generar una nueva pantalla de ranas.

```
2000 REM dibujar la escena
2010 INK 0: PAPER 6
2020 CLS
2030 GO SUB 2600
2040 INK 2: BRIGHT 1
2050 LET AMOUNT=(4*SC):
      GO SUB 2500
2060 IF FOOD=0 THEN
      LET FOOD=INT (RND*10)+15
2070 INK 4: BRIGHT 0:
      LET AMOUNT=FOOD:
       GO SUB 2500
2080 INK 0
2090 PRINT AT 0,0;"Tanteo=";
     STOTAL+C
2100 GO SUB 2200
2110 IF TIME>=RND*2+3 OR
     INKEY$<>"" THEN  RETURN
```

```
2120 GO SUB 2300
2130 GO TO 2110
2200 POKE 23674,0
2210 POKE 23673,0
2220 POKE 23672,0
2230 LET TIME=0
2299 RETURN
2300 LET TIME=(65536*PEEK 23674+
     256*PEEK 23673+
     PEEK 23672)/50
2310 RETURN
```

La llamada a la subrutina 2600 produce una serpiente de tamaño fijo, y que empieza cada vez en la misma posición. La subrutina es la responsable de inicializar las coordenadas de las matrices X e Y, y los punteros Q y Z. La subrutina 2500 imprime los sapos y las ranas dependiendo del color de la tinta anteriormente asignado. La línea 2040 se utiliza para dar sapos, y la 2070 ranas. El número de ranas aumenta cada vez que acabamos con todas las ranas de la pantalla anterior, y esto hace el juego cada vez más difícil. El número de ranas está dado por FOOD, y va disminuyendo desde su valor inicial, según vamos comiendo ranas. Cuando llega a cero, la línea 2060 genera un número aleatorio de ranas para llenar la pantalla siguiente. La parte final de la subrutina imprime la puntuación final, y espera para volver a empezar durante un período de tiempo también aleatorio, o hasta que se pulse una tecla.

La subrutina 2600 parece algo más complicada, porque no sólo dibuja la serpiente, sino que también inicializa las coordenadas de las matrices X e Y:

```
2600 REM dibujar la serpiente
2610 INK 0: PAPER 6
2620 LET X=2: LET Y=10
2625 LET XH=X: LET YH=10
2630 LET H$="[A]"
2640 LET S$="[B]"
2650 LET E$="[C]"
2660 PRINT AT Y,X;E$;
2670 LET X(1)=X: LET Y(1)=Y
2680 FOR Q=2 TO 10
2690 LET X=X+1
2700 PRINT AT Y,X;S$;
2710 LET X(Q)=X: LET Y(Q)=Y
2720 NEXT Q
2730 LET X=X+1
2740 PRINT AT Y,X;H$;
2750 LET X(11)=X: LET Y(11)=Y
```

```
2760 LET Z=1: LET Q=11
2770 LET XV=1: LET YV=0
2799 RETURN
```

Las variables X, Y y XH, YH se emplean para almacenar la posición actual de la cabeza de la serpiente. La razón de utilizar dos pares de variables se hace patente en la subrutina 4500 que más tarde veremos. H$ y S$ guardan los caracteres de la cabeza y cuerpo de la serpiente actual, y E$ guarda el carácter de la cola. La línea 2660 imprime el carácter de la cola en 10, 2, y luego la línea 2670 almacena sus coordenadas en X(1), Y(1). El bucle FOR (líneas 2680 a 2720) imprime los caracteres que forman el cuerpo, y almacena sus coordenadas en X, Y. Finalmente, las líneas 2730 a 2750 imprimen la cabeza, y las líneas 2760 y 2770 inicializan las variables que controlan la longitud de la serpiente y la dirección inicial de movimiento. La variable Z apunta a las coordenadas actuales de la cola en X, Y, y Q apunta a las de la cabeza. Las variables XV e YV controlan la dirección del movimiento y serán explicadas con más detalle en las subrutinas 4000 y 4500.

La subrutina 2500 es muy simple:

```
2500 REM dibujar la comida
2505 IF AMOUNT=0 THEN  RETURN
2510 FOR K=1 TO AMOUNT
2520 LET U=RND*30-1
2530 LET V=RND*20+1
2540 IF SCREEN$ (V,U)<>" " THEN
      GO TO 2520
2550 PRINT AT V,U;"[F]";
2560 NEXT K
2599 RETURN
```

El bucle FOR (líneas 2510 a 2560) imprimirá AMOUNT ranas o sapos en lugares al azar. Serán ranas o sapos dependiendo del color de la tinta asignado antes de llamar a la subrutina.

Las líneas 2520 y 2530 generan números aleatorios que se utilizarán como coordenadas potenciales para un sapo o una rana. Si ya hubiese un carácter en V, U, la línea 2540 transfiere otra vez el control, y otros dos números aleatorios son generados. La comprobación de si existe o no un carácter, se hace por medio de la función SCREEN$ aplicada en V, U, que nos da como resultado el carácter colocado en dicha posición.

Subrutina 3000 y subrutinas asociadas

La subrutina 3000 solamente llama a otras dos subrutinas para hacer todo lo necesario para mover la serpiente:

```
3000 REM un movimiento
3010 GO SUB 4000
3020 GO SUB 4500
3099 RETURN
```

La subrutina 4000 examina el teclado para saber qué tecla de las flechas es pulsada, y luego actualiza las variables que indican la dirección del movimiento de la serpiente. El movimiento actual es producido por la subrutina 4500. Esta subrutina no sólo actualiza las coordenadas de las matrices e imprime los caracteres necesarios, sino que también comprueba si la serpiente ha chocado contra algo.

La subrutina 4000 es muy similar a otras subrutinas de teclado empleadas en programas anteriores:

```
4000 REM conducir
4010 IF INKEY$="6" THEN
      LET XV=0: LET YV=1:
       LET S$="[J]":
        LET H$="[I]"
4020 IF INKEY$="7" THEN
      LET XV=0: LET YV=-1:
       LET S$="[H]":
        LET H$="[G]"
4030 IF INKEY$="5" THEN
     LET XV=-1: LET YV=0:
      LET S$="[E]":
       LET H$="[D]"
4040 IF INKEY$="8" THEN
      LET XV=1: LET YV=0:
       LET S$="[B]":
        LET H$="[A]"
4050 BEEP .01,-10
4099 RETURN
```

La función INKEY$ se utiliza para llevar el control de la flecha en turno (líneas 4010 a 4040). Si una determinada flecha se pulsa, las variables XV, YV toman los valores que corresponden a esa determinada dirección. Por ejemplo, si pulsamos la flecha derecha, la variable XV toma el valor 1, e YV el valor 0, de modo que al sumar XV, YV a X, Y (actual posición de la cabeza), se produce claramente un movimiento a la derecha. Del mismo modo que la velocidad queda establecida en respuesta a la presión de una flecha, se ha de determinar el carácter apropiado para la cabeza y el cuerpo. Por ejemplo, si se pulsa la flecha derecha, la línea 4040 asigna un carácter de cuerpo hacia la derecha a la variable S$, y una cabeza derecha a H$. La última parte de la subrutina consigue un sonido siseante empleando BEEP.

La subrutina 4500 es probablemente la subrutina más complicada de todo el programa:

```
4500 REM actualizar
4510 LET XH=X: LET YH=Y
4520 LET X=X+XV: LET Y=Y+YV
4530 IF X>31 THEN  LET X=0
4540 IF X<0 THEN  LET X=31
4550 IF Y>21 THEN  LET Y=1
4560 IF Y<1 THEN  LET Y=21
4570 IF SCREEN$ (Y,X)<>" " THEN
      GO SUB 5000
4580 PRINT AT YH,XH;S$;
4590 PRINT AT Y,X;H$;
4600 LET Q=Q+1
4610 IF Q>MAX THEN  LET Q=1
4620 LET X(Q)=X: LET Y(Q)=Y
4630 PRINT AT Y(Z),X(Z);" ";
4640 LET Z=Z+1
4650 IF Z>MAX THEN  LET Z=1
4660 PRINT AT Y(Z),X(Z);E$;
4699 RETURN
```

La línea 4510 almacena la actual posición de la cabeza dada por Y, X en YH, XH, después la línea 4520 actualiza la posición de la cabeza añadiendo las velocidades YV y XV, posteriormente las líneas 4530 a 4560 se aseguran de que la nueva posición todavía está dentro de la pantalla. Hay dos formas de enfrentarse a esta situación de que la serpiente llegue al borde de la pantalla; se puede tratar como una falta, y hacer que la serpiente pierda una vida, o pensar que los bordes de la pantalla están pegados el uno al otro, es decir, que cuando la serpiente desaparece por el borde superior de la pantalla, vuelve a aparecer por el borde inferior. Esto se conoce como «Universo esférico», y será la estrategia usada en este juego.

La línea 4570 utiliza SCREEN$ para comprobar si la serpiente ha chocado contra algo; si es así, la subrutina 5000 es llamada para hacerse cargo de la situación. En caso contrario, la línea 4580 imprime un carácter de cuerpo en la antigua posición de la cabeza que así queda borrada, y la línea 4590 imprime la cabeza en su nueva posición. Observamos que la rutina 4500 no tiene que preocuparse de cuál de todos los caracteres de la cabeza o el cuerpo tiene que utilizar, ya que los correctos han sido seleccionados por la rutina 4000. De acuerdo con esto, la nueva posición de la cabeza es almacenada en las coordenadas de las matrices X e Y (línea 4620), después de que el puntero de la cabeza Q ha sido actualizado (líneas 4600 y 4610).

La tarea final de la subrutina 4500 es mover la cola de la serpiente.

La línea 4630 borra la cola de su antigua posición. Entonces, las líneas 4640 y 4650 actualizan el puntero de la cola, Z, y la línea 4660 imprime el carácter de la cola en su nueva posición.

La subrutina 5000 y subrutinas asociadas

La subrutina 5000 es la responsable de responder cuando la serpiente se dirige hacia algo:

```
5000 REM acierto
5010 LET COL=ATTR (Y,X)
5020 IF COL=48 OR COL=114 THEN
      GO SUB 5500
5030 IF COL=52 THEN  GO SUB 5600
5099 RETURN
```

Cuando la subrutina 5000 es llamada, sólo sabemos que la serpiente va a chocar contra algo y lo primero que ha de hacer es descubrir contra qué. La línea 5010 utiliza ATTR para descubrir el color del carácter al que se dirige la serpiente. Si es rojo brillante o negro, la serpiente ha chocado con un sapo, o consigo misma, respectivamente, y se llama a la subrutina 5500 (línea 5020) para quitar una vida a la serpiente. Si el color resulta ser verde, se llama a la subrutina 5600 para añadir uno a la puntuación y aumentar en un carácter a la serpiente (línea 5030).

La subrutina 5500 es bastante corta, pero su forma de interactuar en el programa es bastante complicada:

```
5500 REM sapo-a si misma
5505 BEEP .2,-30
5520 LET SN=SN+1
5530 IF SN>4 THEN
      LET STOTAL=STOTAL+C:
       LET NOLIVES=1: RETURN
5540 GO SUB 2000
5599 RETURN
```

La línea 5005 avisa al jugador de que ha perdido una vida, mediante un sonido. Entonces la línea 5520 añade uno al contador SN. Si SN es mayor que 4, se han empleado todas las vidas, y el juego termina. La línea 5530 lo comprueba, y asigna VERDADERO a la variable NOLIVES (es decir 1), para informar al programa principal. En caso contrario, se llama a la subrutina 2000 para redibujar la serpiente, y dejar las ranas y sapos preparados para continuar el juego.

La subrutina 5600 tiene dos tareas que realizar, cuidar del tanteo, y aumentar en un carácter el tamaño de la serpiente:

```
5600 REM rana
5610 BEEP .1,10
5620 LET C=C+1: LET FOOD=FOOD-1
5630 PRINT AT 0,0;"Tanteo=";
     STOTAL+C
5640 LET Z=Z-1
5650 IF Z<1 THEN  LET Z=MAX
5660 IF FOOD<>0 THEN  RETURN
5670 LET STOTAL=STOTAL+C
5680 LET C=0
5690 IF SC=4 THEN  LET NOFOOD=1:
       RETURN
5700 LET SC=SC+1
5710 GO SUB 2000
5799 RETURN
```

La línea 5610 ejecuta un sonido para indicar al jugador que una rana ha sido comida. Luego la línea 5620 añade uno a la puntuación, y resta uno a FOOD. La razón para reducir la variable FOOD en uno, es que ésta se utiliza como indicador de que todas las ranas han sido comidas, y lleva la cuenta del número de ranas que han de ser repintadas si la serpiente pierde una vida y el juego sigue. La línea 5630 imprime la puntuación final.

Aumentar en un carácter la longitud de la rana es muy simple. Las líneas 5640 y 5650 ajustan el puntero de la Z, y, por tanto, mueven la posición de la cola un lugar. Cuando la subrutina 5600 termina, el control es devuelto a la subrutina 4500, que está moviendo la serpiente hacia delante, y, gracias a esto, el efecto de ajustar consiste en dejar la cola en su posición, aunque la cabeza se mueva. Por tanto, la longitud de la serpiente aumenta automáticamente al moverse.

La parte final de la subrutina 5600 detecta el momento en que todas las ranas han sido comidas. Si es así, la línea 5670 añade C al total. La línea 5690 comprueba si todas las pantallas de ranas se han acabado, y en este caso asigna VERDADERO a NOFOOD (es decir 1). Si aún quedan pantallas con ranas, se aumenta SC, y la subrutina 2000 es llamada para ver si FOOD es cero, y, en caso de serlo, genera un nuevo número de ranas al azar.

Subrutina 7000

El final de este juego no es demasiado espectacular, ya que termina imprimiendo simplemente un mensaje, que dice cómo se ha jugado teniendo en cuenta el número de pantallas de ranas que se han comido antes de perder las cinco vidas.

Si se han acabado todas las pantallas, las palabras «SUPER SERPIENTE» se imprimen del mismo modo que el título:

```
7000 REM fin del juego
7010 CLS
7020 INK 0
7030 IF SC=0 THEN
      PRINT AT 10,3;
      "Degradada a lucion"
7040 IF SC=1 THEN
      PRINT AT 10,3;
      "Las ranas no tienen";
      AT 11,3;"de que ";
      "preocuparse"
7050 IF SC=2 THEN
      PRINT AT 10,3;
      "No esta mal";
      AT 11,3;"para una ";
      "culebrilla"
7060 IF SC=3 THEN
      PRINT AT 10,3;
      "Bien reptado"
7070 IF SC=4 THEN
      PRINT AT 10,3;
      "Una venenosa actuacion"
7075 IF SC=4 AND NOFOOD THEN
      GO SUB 7500
7077 LET N=0
7080 FOR I=5 TO 1 STEP -1
7090 IF STOTAL>=T(I) THEN
      LET N=I
7100 NEXT I
7105 PRINT
7110 IF N=0 THEN  PRINT TAB 3;
      "Tu tanteo ";STOTAL:
       GO TO 7310
7120 PRINT TAB 3;
     "Ahora estas ";
     "en el puesto ";N
7130 PRINT TAB 3;
     "de la liga serpiente"
7140 IF N=5 THEN  GO TO 7200
7150 FOR I=5 TO N+1 STEP -1
7160 LET T(I)=T(I-1)
7170 LET T$(I)=T$(I-1)
```

```
7180 NEXT I
7200 LET T(N)=STOTAL
7210 INPUT "Cual es tu nombre ?"
     ;T$(N)
7220 CLS
7230 PRINT "[G CBBBBBBBBBBBB]";
     "[BBBBBBBBBBBBBA C]"
7240 FOR I=1 TO 20
7250 PRINT AT I,0;"[H]";
     AT I,31;"[J]"
7260 NEXT I
7270 PRINT "[C DEEEEEEEEEEEE]";
     "[EEEEEEEEEEC I]"
7280 FOR I=1 TO 5
7290 PRINT AT I*2+5,2;T$(I);
     TAB (20);T(I);
7300 NEXT I
7310 INPUT "Otra reptada s/n ?";
     A$
7320 IF A$<>"s" AND A$<>"n" THEN
      GO TO 7310
7330 LET AGAIN=0
7340 IF A$="s" THEN  LET AGAIN=1
7399 RETURN
7505 RESTORE 7510
7510 DATA 8,2,2,1,0,3,0,0,7,0,8,
     2,2,1
7520 DATA 0,8,2,2,1,0,8,2,2,1
7530 DATA 8,0,0,0,0,10,0,0,8,0,
     8,0,0,10
7540 DATA 0,8,0,0,0,0,8,0,0,10
7550 DATA 5,5,5,8,0,10,0,0,8,0,
     8,4,5,10
7560 DATA 0,8,5,8,0,0,10,5,5,8
7570 DATA 0,0,0,8,0,10,0,0,8,0,
     8,0,0,0
7580 DATA 0,8,0,0,0,0,8,10,2,0
7590 DATA 3,2,2,2,0,10,2,2,2,0,
     3,0,0,0,0
7600 DATA 5,5,5,3,0,3,0,0,9
7602 PAUSE 20
7604 CLS
7610 FOR I=1 TO 5
7620 PRINT TAB 2;
7630 FOR J=1 TO 24
```

```
7640 READ S
7650 IF S=0 THEN  PRINT " ";:
       GO TO 7670
7660 PRINT CHR$ (143+S);
7670 NEXT J
7680 PRINT
7690 NEXT I
7695 GO SUB 8500
7699 RETURN
```

Las líneas 7030 a 7070 imprimen los mensajes de felicitación. Si NO-FOOD es verdad y SC = 4, entonces las líneas 7500 a 7699 imprimen la palabra «SUPER», del mismo modo que la palabra «SERPIENTE» era impresa en la subrutina 8500. Luego es llamada dicha subrutina para imprimir la palabra «SERPIENTE».

Una característica adicional del final del juego, es que lleva la cuenta de la liga de los resultados. Si el jugador está entre los cinco mejores, le pregunta el nombre, y lo coloca dentro de la tabla en la posición correcta. Las líneas 7080 a 7100 encuentran el lugar que corresponde a dicha puntuación, buscando la primera puntuación mayor o igual en la tabla. Si el jugador quiere ser colocado en la liga, todas las puntuaciones mayores o iguales bajan un puesto (líneas 7150 a 7180), y el nombre y la puntuación correspondiente quedan colocados (líneas 7200 y 7210). Las líneas 7220 a 7270 rodean la pantalla con cuatro serpientes, la tabla se imprime (7280 a 7300), y se pregunta al jugador si desea volver a jugar (líneas 7310 a 7399). Es conveniente observar que para volver a empezar no se debe pulsar RUN, ya que se borraría el contenido de las matrices que llevan cuenta de la liga,y, por lo tanto, utilizamos la variable AGAIN para indicar que el resto del programa se ha de volver a ejecutar.

Conclusión - Cómo jugar

Cualquier juego en el que participe una serpiente resulta muy divertido. Esto no quiere decir que no haya campo suficiente para mejoras e innovaciones, ya que el éxito del juego se debe a que cuenta con la velocidad y el interés que sólo se encuentra en juegos escritos en Ensamblador. En el siguiente capítulo se explorará en otro juego el gran potencial del uso de la serpiente.

La versión final —Listado completo

```
  10 REM serpiente
  20 CLS
  30 LET AGAIN=0: REM falso
  40 GO SUB 1000
  50 GO SUB 8000
  60 GO SUB 2000
  70 REM bucle
  80 GO SUB 3000
  90 IF NOFOOD OR NOLIVES THEN
      GO TO 110
 100 GO TO 70
 110 GO SUB 7000
 120 IF AGAIN THEN  GO TO 40
 130 STOP


1000 REM inicializacion
1005 RESTORE 1000
1010 DATA "A",56,124,250,255,
     255,250,124,56
1020 DATA "B",126,239,247,251,
     251,247,239,126
1030 DATA "C",24,60,126,255,
     255,126,60,24
1040 DATA "D",28,62,95,255,
     255,95,62,28
1050 DATA "E",126,247,239,223,
     223,239,247,126
1060 DATA "F",130,84,56,60,
     254,130,68,130
1070 DATA "G",24,60,90,255,
     255,255,126,60
1080 DATA "H",60,255,231,219,
     189,255,255,126
1090 DATA "I",60,126,255,255,
     255,90,60,24
1100 DATA "J",126,255,255,189,
     219,231,255,126
1110 DIM C$(1)
1120 FOR I=1 TO 10
1130 READ C$
1140 FOR J=0 TO 7
1145 READ D
1150 POKE USR C$+J,D
1160 NEXT J
```

```
1170 NEXT I
1190 IF AGAIN THEN  GO TO 1260
1200 DIM X(50): DIM Y(50)
1205 LET MAX=50
1210 DIM T$(5,15)
1220 DIM T(5)
1230 FOR I=1 TO 5
1240 LET T(I)=0
1250 NEXT I
1260 LET NOFOOD=0
1270 LET NOLIVES=0
1280 LET HIT=0
1290 LET C=0: LET STOTAL=0
1300 LET SC=0: LET SN=0:
      LET FOOD=0
1399 RETURN
2000 REM dibujar la escena
2010 INK 0: PAPER 6
2020 CLS
2030 GO SUB 2600
2040 INK 2: BRIGHT 1
2050 LET AMOUNT=(4*SC):
      GO SUB 2500
2060 IF FOOD=0 THEN
      LET FOOD=INT (RND*10)+15
2070 INK 4: BRIGHT 0:
      LET AMOUNT=FOOD:
       GO SUB 2500
2080 INK 0
2090 PRINT AT 0,0;"Tanteo=";
     STOTAL+C
2100 GO SUB 2200
2110 IF TIME>=RND*2+3 OR
     INKEY$<>"" THEN  RETURN
2120 GO SUB 2300
2130 GO TO 2110
2200 POKE 23674,0
2210 POKE 23673,0
2220 POKE 23672,0
2230 LET TIME=0
2299 RETURN
2300 LET TIME=(65536*PEEK 23674+
     256*PEEK 23673+
     PEEK 23672)/50
2310 RETURN
```

```
2500 REM dibujar la comida
2505 IF AMOUNT=0 THEN  RETURN
2510 FOR K=1 TO AMOUNT
2520 LET U=RND*30-1
2530 LET V=RND*20+1
2540 IF SCREEN$ (V,U)<>" " THEN
      GO TO 2520
2550 PRINT AT V,U;" ";
2560 NEXT K
2599 RETURN

2600 REM dibujar la serpiente
2610 INK 0: PAPER 6
2620 LET X=2: LET Y=10
2625 LET XH=X: LET YH=10
2630 LET H$=" "
2640 LET S$=" "
2650 LET E$=" "
2660 PRINT AT Y,X;E$;
2670 LET X(1)=X: LET Y(1)=Y
2680 FOR Q=2 TO 10
2690 LET X=X+1
2700 PRINT AT Y,X;S$;
2710 LET X(Q)=X: LET Y(Q)=Y
2720 NEXT Q
2730 LET X=X+1
2740 PRINT AT Y,X;H$;
2750 LET X(11)=X: LET Y(11)=Y
2760 LET Z=1: LET Q=11
2770 LET XV=1: LET YV=0
2799 RETURN

3000 REM un movimiento
3010 GO SUB 4000
3020 GO SUB 4500
3099 RETURN


4000 REM conducir
4010 IF INKEY$="6" THEN
      LET XV=0: LET YV=1:
       LET S$=" ": LET H$=" "
4020 IF INKEY$="7" THEN
      LET XV=0: LET YV=-1:
       LET S$=" ": LET H$=" "
```

```
4030 IF INKEY$="5" THEN
      LET XV=-1: LET YV=0:
       LET S$=" ": LET H$=" "
4040 IF INKEY$="8" THEN
      LET XV=1: LET YV=0:
       LET S$=" ": LET H$=" "
4050 BEEP .01,-10
4099 RETURN


4500 REM actualizar
4510 LET XH=X: LET YH=Y
4520 LET X=X+XV: LET Y=Y+YV
4530 IF X>31 THEN  LET X=0
4540 IF X<0 THEN  LET X=31
4550 IF Y>21 THEN  LET Y=1
4560 IF Y<1 THEN  LET Y=21
4570 IF SCREEN$ (Y,X)<>" " THEN
      GO SUB 5000
4580 PRINT AT YH,XH;S$;
4590 PRINT AT Y,X;H$;
4600 LET Q=Q+1
4610 IF Q>MAX THEN  LET Q=1
4620 LET X(Q)=X: LET Y(Q)=Y
4630 PRINT AT Y(Z),X(Z);" ";
4640 LET Z=Z+1
4650 IF Z>MAX THEN  LET Z=1
4660 PRINT AT Y(Z),X(Z);E$;
4699 RETURN


5000 REM acierto
5010 LET COL=ATTR (Y,X)
5020 IF COL=48 OR COL=114 THEN
      GO SUB 5500
5030 IF COL=52 THEN  GO SUB 5600
5099 RETURN


5500 REM sapo-a si misma
5505 BEEP .2,-30
5520 LET SN=SN+1
5530 IF SN>4 THEN
      LET STOTAL=STOTAL+C:
       LET NOLIVES=1: RETURN
```

```
5540 GO SUB 2000
5599 RETURN


5600 REM rana
5610 BEEP .1,10
5620 LET C=C+1: LET FOOD=FOOD-1
5630 PRINT AT 0,0;"Tanteo=";
     STOTAL+C
5640 LET Z=Z-1
5650 IF Z<1 THEN  LET Z=MAX
5660 IF FOOD<>0 THEN  RETURN
5670 LET STOTAL=STOTAL+C
5680 LET C=0
5690 IF SC=4 THEN  LET NOFOOD=1:
       RETURN
5700 LET SC=SC+1
5710 GO SUB 2000
5799 RETURN


7000 REM fin del juego
7010 CLS
7020 INK 0
7030 IF SC=0 THEN  PRINT AT 10,3
;      "Degradada a lucion"
7040 IF SC=1 THEN  PRINT AT 10,3
;      "Las ranas no tienen";
       AT 11,3;"de que ";
       "preocuparse"
7050 IF SC=2 THEN  PRINT AT 10,3
;      "No esta mal";
       AT 11,3;"para una ";
       "culebrilla"
7060 IF SC=3 THEN  PRINT AT 10,3
;      "Bien reptado"
7070 IF SC=4 THEN  PRINT AT 10,3
;      "Una venenosa actuacion"
7075 IF SC=4 AND NOFOOD THEN
       GO SUB 7500
7077 LET N=0
7080 FOR I=5 TO 1 STEP -1
7090 IF STOTAL>=T(I) THEN
       LET N=I
```

```
7100 NEXT I
7105 PRINT
7110 IF N=0 THEN  PRINT TAB 3;
      "Tu tanteo ";STOTAL:
       GO TO 7310
7120 PRINT TAB 3;
     "Ahora estas ";
     "en el puesto ";N
7130 PRINT TAB 3;
     "de la liga serpiente"
7140 IF N=5 THEN  GO TO 7200
7150 FOR I=5 TO N+1 STEP -1
7160 LET T(I)=T(I-1)
7170 LET T$(I)=T$(I-1)
7180 NEXT I
7200 LET T(N)=STOTAL
7210 INPUT "Cual es tu nombre ?"
     ;T$(N)
7220 CLS
7230 PRINT "                    ";
     "                "
7240 FOR I=1 TO 20
7250 PRINT AT I,0;" ";AT I,31;
     " "
7260 NEXT I
7270 PRINT "                    ";
     "                "
7280 FOR I=1 TO 5
7290 PRINT AT I*2+5,2;T$(I);
     TAB (20);T(I);
7300 NEXT I
7310 INPUT "Otra reptada s/n ?";
     A$
7320 IF A$<>"s" AND A$<>"n" THEN
      GO TO 7310
7330 LET AGAIN=0
7340 IF A$="s" THEN  LET AGAIN=1
7399 RETURN
7505 RESTORE 7510
7510 DATA 8,2,2,1,0,3,0,0,7,0,8,
     2,2,1
7520 DATA 0,8,2,2,1,0,8,2,2,1
7530 DATA 8,0,0,0,0,10,0,0,8,0,
     8,0,0,10
```

```
7540 DATA 0,8,0,0,0,0,8,0,0,10
7550 DATA 5,5,5,8,0,10,0,0,8,0,
     8,4,5,10
7560 DATA 0,8,5,8,0,0,10,5,5,8
7570 DATA 0,0,0,8,0,10,0,0,8,0,
     8,0,0,0
7580 DATA 0,8,0,0,0,0,8,10,2,0
7590 DATA 3,2,2,2,0,10,2,2,2,0,
     3,0,0,0,0
7600 DATA 5,5,5,3,0,3,0,0,9
7602 PAUSE 20
7604 CLS
7610 FOR I=1 TO 5
7620 PRINT TAB 2;
7630 FOR J=1 TO 24
7640 READ S
7650 IF S=0 THEN  PRINT " ";:
       GO TO 7670
7660 PRINT CHR$ (143+S);
7670 NEXT J
7680 PRINT
7690 NEXT I
7695 GO SUB 8500
7699 RETURN


8000 REM titulo
8005 INK 0: PAPER 6
8006 CLS
8007 BORDER 6
8010 GO SUB 8500
8020 PRINT AT 9,1;
8030 PRINT "Debes guiar tu ";
     "serpiente usando"
8040 PRINT "las cuatro teclas ";
     "de cursor, y"
8050 PRINT "comerte todas las ";
     "ranas verdes ";
8060 PRINT  INK 4;" "
8070 PRINT
8080 PRINT "Si tratas de ";
     "comerte un sapo ";
8090 PRINT  BRIGHT 1; INK 2;" "
8100 PRINT " o a ti misma"
```

```
8110 PRINT "entonces perderas ";
     "una de tus"
8120 PRINT "cinco vidas"
8130 PRINT
8140 PRINT "has de comerte ";
     "cinco pantallas"
8150 PRINT "antes de ";
     "convertirte"
8160 PRINT "en una  ";
     "SUPER SERPIENTE"
8170 GO SUB 2200
8180 GO SUB 2300
8190 IF TIME<=50 AND INKEY$=""
      THEN  GO TO 8180
8199 RETURN


8500 REM serpiente
8505 PRINT : PRINT
8510 DATA 8,2,2,1,0,8,2,2,2,0,8,
     2,2,2,0
8520 DATA 3,0,8,1,0,8,2,2,1
8530 DATA 8,0,0,0,0,8,0,0,10,0,
     8,0,0,10
8540 DATA 0,10,8,2,0,0,8,0,0,0
8550 DATA 5,5,5,8,0,8,0,0,10,0,
     8,5,5
8560 DATA 10,0,10,8,0,0,0,5,5,8,
     0
8570 DATA 0,0,0,8,0,8,0,0,10,0,
     8,10,2,2
8580 DATA 0,10,5,8,0,0,8,2,2,0
8583 DATA 0,0,0,8,0,8,0,0,10,0,
     8,0,0
8586 DATA 10,0,10,0,5,8,0,8,0,0,
     0
8590 DATA 3,2,2,2,0,3,0,0,9,0,3,
     0,0,9
8595 DATA 0,9,0,0,3,0,5,5,5,3
8600 RESTORE 8500
8610 FOR I=1 TO 6
8620 PRINT TAB 2;
8630 FOR J=1 TO 24
```

```
8640 READ S
8650 IF S=0 THEN  PRINT " ";:
       GO TO 8670
8660 PRINT CHR$ (143+S);
8670 NEXT J
8680 PRINT
8690 NEXT I
8699 RETURN
```

6. Los Renacuajos

La serpiente animada descrita en el capítulo anterior, puede ser utilizada para crear otros juegos igualmente divertidos. «Los renacuajos» es uno de ellos, y en este capítulo se da una demostración de cómo el programa de «La serpiente» se puede modificar y dar lugar a un juego nuevo.

Diseño del juego

La idea básica de «Los renacuajos» es muy similar a la desarrollada en el juego anterior; una serpiente tiene que comer a unas ranas para ganar puntos. La única diferencia es que, en este caso, las ranas también se mueven. Al principio del juego las ranas son todavía renacuajos que nadan en una charca situada en el centro de la pantalla. Uno a uno se van convirtiendo en ranas, y rápidamente se dirigen hacia el borde de la pantalla. La serpiente no puede comerse los renacuajos hasta que se han convertido en ranas, y pierde una vida si cae dentro de la charca. El objeto del juego es comer el mayor número posible de ranas, antes de que éstas escapen por el borde de la pantalla.

Aparte del pozo y las ranas, móviles, el resto de los elementos del juego es igual que en el juego original de la serpiente. La serpiente tiene cinco vidas, y ha de comer cinco pantallas de ranas. Si la serpiente pierde una vida, milagrosamente las ranas que quedan se vuelven renacuajos, y la serpiente tiene otra oportunidad de caer mientras espera. Todos los caracteres gráficos utilizados en este juego han sido introducidos como parte del anterior, y aparte de la charca,la forma de la pantalla es muy similar, exceptuando que en «La serpiente» las ranas son éstaticas, mientras que en «Los renacuajos» las ranas se mueven.

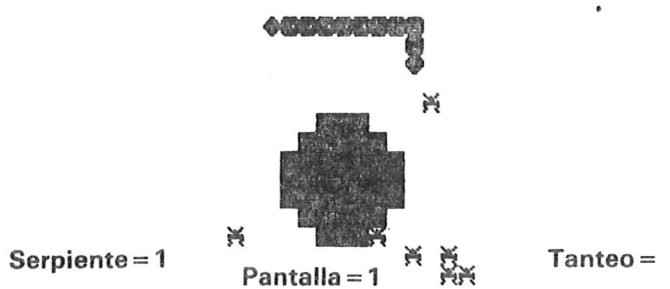


Figura 6.1.

Puesta en escena

Las modificaciones se hacen en dos partes: primero los cambios y ampliaciones necesarios para establecer el juego, y después los cambios y ampliaciones necesarios para animar las ranas. En esta sección trataremos la primera parte, y la segunda se tratará en la siguiente sección.

La mayor parte de las modificaciones necesarias para establecer el juego se encuentran en una subrutina, la subrutina 2000. Ahora, en lugar de imprimir ranas y sapos, tiene que imprimir una charca llena de renacuajos en el centro de la pantalla.

Es mejor imprimir la charca por medio de una nueva subrutina (2500) y los renacuajos por medio de una modificación de la subrutina 2500. La nueva versión de la subrutina 2000 es:

```
2000 REM dibujar la escena
2010 INK 0: PAPER 6
2020 CLS
2030 GO SUB 2600
2040 GO SUB 2150
2050 INK 4
2060 IF FOOD=0 THEN
      LET FOOD=NUM
2070 LET AMOUNT=FOOD:
      GO SUB 2500
2080 INK 0
2090 PRINT AT 0,0;"CULEBRA=";
     SN+1;" PANTALLA=";SC+1;
     " PUNTOS=";STOTAL+C
2095 PLOT 0,167: DRAW 255,0
2100 GO SUB 2200
2110 IF TIME>=RND*2+3 OR
     INKEY$<>"" THEN  RETURN
2120 GO SUB 2300
2130 GO TO 2100
```

La nueva subrutina que dibuja la charca es llamada por la línea 2040. La subrutina 2500 sólo es llamada una vez (línea 2070), para imprimir un número fijo de renacuajos dentro de la charca. Los únicos cambios que quedan son añadir la línea 2090 para imprimir la serpiente y el número de pantalla y la inclusión de los comandos de gráficos en alta resolución en la línea 2095 para dibujar la línea horizontal que marca el límite del área del juego.

La subrutina 2150 no es más que una lista de sentencias CIRCLE que imprime una charca utilizando color cyan para el papel y la tinta.

```
2150 REM dibujar la charca
2170 FOR P=1 TO 27 STEP 2
2180 CIRCLE  INK 5; PAPER 5;
      127,87,P
2190 NEXT P
2195 INK 0: PAPER 6
2199 RETURN
```

Utilizamos cyan en lugar de azul, porque tiene un contraste mejor con el resto de los colores en un aparato de blanco y negro.

La subrutina 2500 coloca los renacuajos en la charca, en lugar de colocar las ranas en cualquier parte de la pantalla. Este arreglo ha sido muy fácil, ya que ha bastado con restringir el rango de los números aleatorios generados en las líneas 2520 y 2530. Más tarde los renacuajos y las ranas que resultan de ellos van a ser animados, y esto implica que sus posiciones son conocidas. Para llevar la cuenta de dónde están todos los renacuajos y las ranas, es necesario emplear un par de matrices para almacenar las coordenadas X, Y de cada renacuajo/rana:

```
2500 REM dibujar la comida
2505 IF AMOUNT=0 THEN  RETURN
2510 FOR K=1 TO AMOUNT
2520 LET U=INT (RND*7)+12
2530 LET V=INT (RND*20)+1
2540 IF ATTR (V,U)<>45
     THEN  GO TO 2520
2550 PRINT AT V,U; INK 0;
      PAPER 5;",";
2555 LET F(K)=U: LET G(K)=V
2560 NEXT K
2570 IF AMOUNT=NUM THEN  RETURN
2580 FOR K=AMOUNT+1 TO NUM
2585 LET F(K)=-1
2586 NEXT K
2599 RETURN
```

La línea 2555 guarda las coordenadas en F, G para uso posterior. El número máximo de renacuajos/ranas es ocho, pero a lo largo del programa van disminuyendo porque escapan o son comidos. Para indicar que uno de ellos ya no está en el programa, se asigna —1 a su coordenada X. El bucle FOR, líneas 2580 y 2586 asignan también —1 a las entradas no utilizadas de la matriz F. Por supuesto que hemos de dimensionar el nuevo par de matrices, y hay que añadir las líneas 1206 y 1207 a la sección inicial:

```
1206 LET NUM=8
1207 DIM F(NUM): DIM G(NUM)
```

Haciendo una escapada

Una vez establecida la puesta en escena, con las coordenadas de todos los renacuajos guardadas en el par de matrices F y G, podemos empezar con la animación de la serpiente y las ranas. El único cambio en el bucle de animación es la adición de la llamada a la subrutina 3500 que mueve una única rana aleatoriamente. La subrutina 3000 queda así:

```
3000 REM un movimiento
3005 IF RND<(.1+.1*SC)
      THEN  GO SUB 3500
3010 GO SUB 4000
3020 GO SUB 4500
3099 RETURN
```

La subrutina 3500 es algo más complicada de lo que cabía esperar. Los principales problemas están en asegurarse de que la rana no aterriza sobre la serpiente, utilizando el color del fondo correcto para imprimir, borrando la rana, y detectando cuándo una rana ha conseguido escapar. La mayor parte de las soluciones a estos problemas han sido introducidas en capítulos anteriores:

```
3500 REM mover la rana
3510 LET K=INT (RND*NUM)+1
3520 IF F(K)=-1 THEN  RETURN
3530 LET N=ATTR (G(K),F(K))
3535 IF N=40 THEN  PAPER 5
3536 IF N<>45 THEN
      PRINT AT G(K),F(K);" ";
3540 LET F(K)=F(K)-SGN (15-F(K))
3550 LET G(K)=G(K)-
                  SGN (11.5-G(K))
3560 IF F(K)<0 OR F(K)>31
      THEN  GO SUB 3900: RETURN
3570 IF G(K)<1 OR G(K)>21
      THEN  GO SUB 3900: RETURN
3574 PAPER 6
3575 LET N$=SCREEN$ (G(K),F(K))
3576 IF N$<>" " THEN  GO TO 3540
3590 PRINT AT G(K),F(K); INK 4;
     "[F]";
3599 RETURN
```

La línea 3510 selecciona una rana/renacuajo para moverlo al azar. Si la coordenada X de éste es —1, es que ya ha sido comido o ha escapado, y, por tanto, la línea 3520 devuelve el control a la subrutina 3000. La línea 3530 encuentra el color del carácter de la antigua posición de la rana, y lo almacena en N. Siempre que este color no sea cyan, la línea 3536

borra la antigua posición. Las líneas 3540 y 3550 actualizan las coordenadas de la rana de modo que ésta se mueva hacia el borde de la pantalla. Es fácil entender como actúa esto, sabiendo que SGN(15—F(K)) es +1 si la rana está a la izquierda del centro de la charca, y -1 si está a la derecha. Así, al restar SGN(15—(K)) de la actual coordenada X de la rana, resulta que la misma se aleja siempre del centro de la charca. Un argumento similar nos muestra que al restar SGN(11.5—G(K)) de la actual coordenada Y de la rana hace que también se aleje del centro de la charca. Siguiendo esto, las líneas 3560 y 3570 comprueban si la rana se sale de la pantalla, y llaman a la subrutina 3900 si es así. La parte final de la rutina, líneas 3547 a 3590 imprimen la rana en su nueva posición, utilizando el color de fondo correcto.

La subrutina 3500 llama a la subrutina 3900 para tomar nota de que una rana ha escapado:

```
3900 REM la rana escapa
3910 LET F(K)=-1
3920 LET FOOD=FOOD-1
3930 IF FOOD<>0 THEN  RETURN
3935 LET STOTAL=STOTAL+C
3936 LET C=0
3940 IF SC=4 THEN
      LET NOFOOD=1: RETURN
3950 LET SC=SC+1
3960 GO SUB 2000
3999 RETURN
```

La línea 3910 asigna -1 a la coordenada horizontal de la rana para indicar durante el resto del programa que la rana ha escapado. El resto de la subrutina quita uno a FOOD, y restablece el juego con una nueva pantalla de ranas, a menos, claro está, que hayamos terminado ya con las cinco pantallas.

Cazando ranas

Las modificaciones que quedan por hacer son principalmente de la subrutina 5600. Siempre que la serpiente llega a un objeto verde se llama a la subrutina (llega a una rana). En la versión original, todo lo que la subrutina 5600 tenía que hacer era sumar uno a la puntuación, y comprobar si todas las ranas habían sido comidas. Ahora también tiene que quitar la rana de la lista de coordenadas en F y G. Esto se hace añadiendo la llamada a una nueva subrutina, 5800:

```
5655 GO SUB 5800
```

Tenemos la oportunidad de alterar la línea 5630 para imprimir la serpiente y el número de pantallas:

```
5630 PRINT AT 0,0;"CULEBRA=";
     SN+1;" PANTALLA=";SC+1;
     " PUNTOS=";STOTAL+C
```

La posición de la rana a la que ha llegado la serpiente, está almacenada en X, Y. La subrutina 5800 borra de la lista de ranas buscando en F y G una rana con las mismas coordenadas, y asignando -1 a su coordenada X.

```
5800 REM quitar rana
5810 LET K=0
5820 REM bucle
5830 LET K=K+1
5840 IF X<>F(K) OR Y<>G(K)
      THEN  GO TO 5820
5850 LET F(K)=-1
5899 RETURN
```

Los toques finales consisten en cambiar los mensajes de la subrutina 7000:

```
7030 IF STOTAL<10 THEN
      PRINT AT 10,3;"DEGRADADO";
      " A LUCION": GO TO 7075
7040 IF STOTAL<20 THEN
      PRINT AT 10,3;"LAS RANAS";
      " NO TIENEN QUE";AT 11,4;
      " PREOCUPARSE":
       GO TO 7075
7050 IF STOTAL<30 THEN
      PRINT AT 10,3;"NO ESTA ";
      "MAL PARA UNA"'"     ";
      "CULEBRILLA": GO TO 7075
7060 IF STOTAL<40 THEN
      PRINT AT 10,3;"BIEN ";
      "JUGADO"
7070 IF SC<50 THEN
      PRINT AT 10,3;"UN ";
      "VENENOSO RESULTADO"
7075 IF NOFOOD THEN  GO SUB 7500
```

Tal como está escrito, «Los renacuajos» es un juego divertido, pero, para hacerlo más interesante, podemos cambiarlo haciendo que si la serpiente se sale de la pantalla pierda una vida:

```
4505 PAPER 6: INK 0
4510 LET XH=X: LET YH=Y
4520 LET X=X+XV: LET Y=Y+YV
4530 IF X>31 THEN  GO SUB 5000
```

```
4540 IF X<0 THEN  GO SUB 5000
4550 IF Y>21 THEN  GO SUB 5000
4560 IF Y<1 THEN  GO SUB 5000
```

y las instrucciones del título también tienen que cambiar de acuerdo con esto:

```
8070 PRINT
8090 PRINT " NO CAIGAS EN EL ";
      "AGUA "
8100 PRINT "NI TE COMAS A TI ";
     "MISMO"
8110 PRINT "O TE SALGAS DE LA";
     " PANTALLA"
8120 PRINT "O PERDERAS UNA DE ";
     "TUS"
8130 PRINT "CINCO VIDAS."
```

Además de estos cambios, como la serpiente en su actual punto de partida inmediatamente llega a la charca, tenemos que cambiar su posición inicial:

```
2620 LET X=2: LET Y=5
2630 LET H$="[A]"
```

La versión final - Listado completo

```
  10 REM renacuajo
  20 CLS
  30 LET AGAIN=0: REM falso
  40 GO SUB 1000
  50 GO SUB 8000
  60 GO SUB 2000
  70 REM bucle
  80 GO SUB 3000
  90 IF NOFOOD OR NOLIVES
      THEN  GO TO 110
 100 GO TO 70
 110 GO SUB 7000
 120 IF AGAIN THEN  GO TO 40
 130 STOP


1000 REM inicializacion
1005 RESTORE 1000
1010 DATA "A",56,124,250,255,
     255,250,124,56
```

```
1020 DATA "B",126,239,247,251,
     251,247,239,126
1030 DATA "C",24,60,126,255,255,
     126,60,24
1040 DATA "D",28,62,95,255,255,
     95,62,28
1050 DATA "E",126,247,239,223,
     223,239,247,126
1060 DATA "F",130,84,56,60,254,
     130,68,130
1070 DATA "G",24,60,90,255,255,
     255,126,60
1080 DATA "H",60,255,231,219,
     189,255,255,126
1090 DATA "I",60,126,255,255,
     255,90,60,24
1100 DATA "J",126,255,255,189,
     219,231,255,126
1110 DATA C$(1)
1120 FOR I=1 TO 10
1130 READ C$
1140 FOR J=0 TO 7
1145 READ D
1150 POKE USR C$+J,D
1160 NEXT J
1170 NEXT I
1190 IF AGAIN THEN  GO TO 1260
1200 DIM X(50): DIM Y(50)
1205 LET MAX=50
1206 LET NUM=8
1207 DIM F(NUM): DIM G(NUM)
1210 DIM T$(5,15)
1220 DIM T(5)
1230 FOR I=1 TO 5
1240 LET T(I)=0
1250 NEXT I
1260 LET NOFOOD=0
1270 LET NOLIVES=0
1280 LET HIT=0
1290 LET C=0: LET STOTAL=0
1300 LET SC=0: LET SN=0:
      LET FOOD=0
1399 RETURN
```

```
2000 REM dibujar la escena
2010 INK 0: PAPER 6
2020 CLS
2030 GO SUB 2600
2040 GO SUB 2150
2050 INK 4
2060 IF FOOD=0 THEN  LET FOOD=NU
M
2070 LET AMOUNT=FOOD:
      GO SUB 2500
2080 INK 0
2090 PRINT AT 0,0;"CULEBRA=";
     SN+1;" PANTALLA=";SC+1;
     " PUNTOS=";STOTAL+C
2095 PLOT 0,167: DRAW 255,0
2100 GO SUB 2200
2110 IF TIME>=RND*2+3 OR
     INKEY$<>"" THEN  RETURN
2120 GO SUB 2300
2130 GO TO 2110

2150 REM dibujar la charca
2170 FOR P=1 TO 27 STEP 2
2180 CIRCLE  INK 5; PAPER 5;
      127,87,P
2190 NEXT P
2195 INK 0: PAPER 6
2199 RETURN
2200 POKE 23674,0
2210 POKE 23673,0
2220 POKE 23672,0
2230 LET TIME=0
2299 RETURN


2300 LET TIME=(65536*PEEK 23674+
     256*PEEK 23673+PEEK 23672)/
     50
2310 RETURN


2500 REM dibujar la comida
2505 IF AMOUNT=0 THEN  RETURN
2510 FOR K=1 TO AMOUNT
2520 LET U=INT (RND*7)+12
```

```
2530 LET V=INT (RND*20)+1
2540 IF ATTR (V,U)<>45
      THEN  GO TO 2520
2550 PRINT AT V,U; INK 0;
      PAPER 5;",";
2555 LET F(K)=U: LET G(K)=V
2560 NEXT K
2570 IF AMOUNT=NUM THEN  RETURN
2580 FOR K=AMOUNT+1 TO NUM
2585 LET F(K)=-1
2586 NEXT K
2599 RETURN

2600 REM dibujar la culebra
2610 INK 0: PAPER 6
2620 LET X=2: LET Y=5
2625 LET XH=X: LET YH=5
2630 LET H$=" "
2640 LET S$=" "
2650 LET E$=" "
2660 PRINT AT Y,X;E$;
2670 LET X(1)=X: LET Y(1)=Y
2680 FOR Q=2 TO 10
2690 LET X=X+1
2700 PRINT AT Y,X;S$;
2710 LET X(Q)=X: LET Y(Q)=Y
2720 NEXT Q
2730 LET X=X+1
2740 PRINT AT Y,X;H$;
2750 LET X(11)=X: LET Y(11)=Y
2760 LET Z=1: LET Q=11
2770 LET XV=1: LET YV=0
2799 RETURN


3000 REM un movimiento
3005 IF RND<(.1+.1*SC)
      THEN  GO SUB 3500
3010 GO SUB 4000
3020 GO SUB 4500
3099 RETURN


3500 REM mover la rana
3510 LET K=INT (RND*NUM)+1
```

```
3520 IF F(K)=-1 THEN  RETURN
3530 LET N=ATTR (G(K),F(K))
3535 IF N=40 THEN  PAPER 5
3536 IF N<>45 THEN
      PRINT AT G(K),F(K);" ";
3540 LET F(K)=F(K)-SGN (15-F(K))
3550 LET G(K)=G(K)-
                 SGN (11.5-G(K))
3560 IF F(K)<0 OR F(K)>31
      THEN  GO SUB 3900: RETURN
3570 IF G(K)<1 OR G(K)>21
      THEN  GO SUB 3900: RETURN
3574 PAPER 6
3575 LET N$=SCREEN$ (G(K),F(K))
3576 IF N$<>" " THEN  GO TO 3540
3590 PRINT AT G(K),F(K); INK 4;
     "[F]";
3599 RETURN


3900 REM la rana escapa
3910 LET F(K)=-1
3920 LET FOOD=FOOD-1
3930 IF FOOD<>0 THEN  RETURN
3935 LET STOTAL=STOTAL+C
3936 LET C=0
3940 IF SC=4 THEN
      LET NOFOOD=1: RETURN
3950 LET SC=SC+1
3960 GO SUB 2000
3999 RETURN

4000 REM obtener la direccion
4010 IF INKEY$="6" THEN
      LET XV=0: LET YV=1:
       LET S$="[J]":
        LET H$="[I]"
4020 IF INKEY$="7" THEN
      LET XV=0: LET YV=-1:
       LET S$="[H]":
        LET H$="[G]"
4030 IF INKEY$="5" THEN
      LET XV=-1: LET YV=0:
       LET S$="[E]":
        LET H$="[D]"
```

```
4040 IF INKEY$="8" THEN
      LET XV=1: LET YV=0:
       LET S$="[B]":
        LET H$="[A]"
4050 BEEP .01,-10
4099 RETURN


4500 REM actualizacion
4505 PAPER 6: INK 0
4510 LET XH=X: LET YH=Y
4520 LET X=X+XV: LET Y=Y+YV
4530 IF X>31 THEN  GO SUB 5000
4540 IF X<0 THEN  GO SUB 5000
4550 IF Y>21 THEN  GO SUB 5000
4560 IF Y<1 THEN  GO SUB 5000
4570 IF SCREEN$ (Y,X)<>" "
      THEN  GO SUB 5000
4580 PRINT AT YH,XH;S$;
4590 PRINT AT Y,X;H$;
4600 LET Q=Q+1
4610 IF Q>MAX THEN  LET Q=1
4620 LET X(Q)=X: LET Y(Q)=Y
4630 PRINT AT Y(Z),X(Z);" ";
4640 LET Z=Z+1
4650 IF Z>MAX THEN  LET Z=1
4660 PRINT AT Y(Z),X(Z);E$;
4699 RETURN


5000 REM atrapar
5010 LET COL=ATTR (Y,X)
5020 IF COL=48 OR COL=45
      THEN  GO SUB 5500
5030 IF COL=52 THEN  GO SUB 5600
5099 RETURN

5500 REM a si misma
5505 BEEP .2,-30
5520 LET SN=SN+1
5530 IF SN>4 THEN
      LET STOTAL=STOTAL+C:
       LET NOLIVES=1: RETURN
5540 GO SUB 2000
5599 RETURN
```

```
5600 REM rana
5610 BEEP .1,10
5620 LET C=C+1: LET FOOD=FOOD-1
5630 PRINT AT 0,0;"CULEBRA=";
     SN+1;" PANTALLA=";SC+1;
     " PUNTOS=";STOTAL+C
5640 LET Z=Z-1
5650 IF Z<1 THEN  LET Z=MAX
5655 GO SUB 5800
5660 IF FOOD<>0 THEN  RETURN
5670 LET STOTAL=STOTAL+C
5680 LET C=0
5690 IF SC=4 THEN
      LET NOFOOD=1: RETURN
5700 LET SC=SC+1
5710 GO SUB 2000
5799 RETURN


5800 REM quitar rana
5810 LET K=0
5820 REM _bucle
5830 LET K=K+1
5840 IF X<>F(K) OR Y<>G(K)
      THEN  GO TO 5820
5850 LET F(K)=-1
5899 RETURN


7000 REM fin del juego
7010 CLS
7020 INK 0
7030 IF STOTAL<10 THEN
      PRINT AT 10,3;"DEGRADADO "
;     "A LUCION": GO TO 7075
7040 IF STOTAL<20 THEN
      PRINT AT 10,3;"LAS RANAS "
;     "NO TIENEN QUE";AT 11,4;
      " PREOCUPARSE": GO TO 7075
7050 IF STOTAL<30 THEN
      PRINT AT 10,3;"NO ESTA ";
      "MAL PARA UNA"'"     ";
     "CULEBRILLA": GO TO 7075
7060 IF STOTAL<40 THEN
     PRINT AT 10,3;"BIEN ";
```

```
      "JUGADO"
7070 IF SC<50 THEN
      PRINT AT 10,3;"UN ";
      "VENENOSO RESULTADO"
7075 IF NOFOOD THEN  GO SUB 7500
7077 LET N=0
7080 FOR I=5 TO 1 STEP -1
7090 IF STOTAL>=T(I)
      THEN  LET N=I
7100 NEXT I
7105 PRINT
7110 IF N=0 THEN  PRINT TAB 3;
      "PUNTUACION:";STOTAL:
       GO TO 7310
7120 PRINT TAB 3;"AHORA FIGURAS"
7130 PRINT TAB 3;"EN LA ";
     "ASOCIACION DE CULEBRAS"
7140 IF N=5 THEN  GO TO 7200
7150 FOR I=5 TO N+1 STEP -1
7160 LET T(I)=T(I-1)
7170 LET T$(I)=T$(I-1)
7180 NEXT I
7200 LET T(N)=STOTAL
7210 INPUT "CUAL ES TU NOMBRE?";
     T$(N)
7220 CLS
7230 PRINT "                    ";
     "                  "
7240 FOR I=1 TO 20
7250 PRINT AT I,0;" ";AT I,31;
     " "
7260 NEXT I
7270 PRINT "                    ";
    "                  "
7280 FOR I=1 TO 5
7290 PRINT AT I*2+5,2;T$(I);
     TAB 20;T(I);
7300 NEXT I
7310 INPUT "OTRO CULEBREO s/n?";
      A$
7320 IF A$<>"s" AND A$<>"n"
      THEN  GO TO 7310
7330 LET AGAIN=0
7340 IF A$="s" THEN  LET AGAIN=1
7399 RETURN
```

```
7500 PRINT
7505 RESTORE 7510
7510 DATA 8,2,2,1,0,3,0,0,7,0,8,
     2,2,1
7520 DATA 0,8,2,2,1,0,8,2,2,1
7530 DATA 8,0,0,0,0,10,0,0,8,0,
     8,0,0,10
7540 DATA 0,8,0,0,0,0,8,0,0,10
7550 DATA 5,5,5,8,0,10,0,0,8,0,
     8,4,5,10
7560 DATA 0,8,5,8,0,0,10,5,5,8
7570 DATA 0,0,0,8,0,10,0,0,8,0,
     8,0,0,0
7580 DATA 0,8,0,0,0,0,8,10,2,0
7590 DATA 3,2,2,2,0,10,2,2,2,0,
     3,0,0,0,0
7600 DATA 5,5,5,3,0,3,0,0,9
7602 PAUSE 20
7604 CLS
7610 FOR I=1 TO 5
7620 PRINT TAB 2;
7630 FOR J=1 TO 24
7640 READ S
7650 IF S=0 THEN  PRINT " ";:
       GO TO 8670
7660 PRINT CHR$ (143+S);
7670 NEXT J
7680 PRINT
7690 NEXT I
7695 GO SUB 8500
7699 RETURN


8000 REM titulo
8005 INK 0: PAPER 6
8006 CLS
8007 BORDER 6
8010 GO SUB 8500
8020 PRINT AT 9,1;
8030 PRINT "DEBES GUIAR TU ";
     "CULEBRA USANDO"
8040 PRINT "LAS CUATRO TECLAS ";
     "DE CURSOR Y"
8050 PRINT "COMERTE TODAS ";
     "LAS RANAS VERDES ";
```

```
8060 PRINT  INK 4;"[F]"
8070 PRINT
8090 PRINT " NO CAIGAS EN EL ";
      "AGUA "
8100 PRINT "NI TE COMAS A TI ";
     "MISMO"
8110 PRINT "O TE SALGAS DE LA";
     " PANTALLA"
8120 PRINT "O PERDERAS UNA DE ";
     "TUS"
8130 PRINT "CINCO VIDAS."
8140 PRINT "TIENES QUE COMERTE"
8150 PRINT "CINCO PANTALLAS DE";
     " RANAS PARA "
8160 PRINT "LLEGAR A SER SUPER";
     " CULEBRA"
8170 GO SUB 2200
8180 GO SUB 2300
8190 IF TIME<=5 AND INKEY$=""
      THEN  GO TO 8180
8199 RETURN

8500 REM culebra
8505 PRINT : PRINT
8510 DATA 8,2,2,1,0,8,2,2,2,0,8,
     2,2,2,0
8520 DATA 3,0,8,1,0,8,2,2,1
8530 DATA 8,0,0,0,0,8,0,0,10,0,
     8,0,0,10
8540 DATA 0,10,8,2,0,0,8,0,0,0
8550 DATA 5,5,5,8,0,8,0,0,10,0,
     8,5,5
8560 DATA 10,0,10,8,0,0,0,5,5,8,
     0
8570 DATA 0,0,0,8,0,8,0,0,10,0,
     8,10,2,2
8580 DATA 0,10,5,8,0,0,8,2,2,0
8583 DATA 0,0,0,8,0,8,0,0,10,0,
     8,0,0
8586 DATA 10,0,10,0,5,8,0,8,0,0,
     0
8590 DATA 3,2,2,2,0,3,0,0,9,0,3,
     0,0,9
8595 DATA 0,9,0,0,3,0,5,5,5,3
8600 RESTORE 8500
```

```
8610 FOR I=1 TO 6
8620 PRINT TAB 2;
8630 FOR J=1 TO 24
8640 READ S
8650 IF S=0 THEN  PRINT " ";:
       GO TO 8670
8660 PRINT CHR$ (143+S);
8670 NEXT J
8680 PRINT
8690 NEXT I
8699 RETURN
```

7. Serpientes y Escalerillas

En este capítulo se da una nueva vida a un tradicional juego bastante conocido. Los juegos tradicionales son una buena cantera de ideas para los juegos de ordenador. No sólo se pueden escribir de acuerdo con las reglas originales, sino que pueden ofrecer un punto de partida para juegos que sólo se pueden jugar con la ayuda de un computador. Como un ejemplo de esta idea, al final de este capítulo «Serpientes y escalerillas» se ha modificado resultando un juego de acción rápida, que nunca podría jugarse con un simple tablero y unos dados.

Este juego es diferente de los del resto del libro en que aparentemente no es animado. Es cierto que la mayor dificultad del juego es la construcción de los gráficos y el tablero, pero, de todos modos, muchas de las técnicas de animación introducidas en capítulos anteriores siguen siendo útiles.

Diseño del juego

La forma y las reglas de «Serpientes y escalerillas» son conocidas por casi todo el mundo, y las únicas cuestiones del diseño del juego conciernen al modo de transferirlas al ordenador. La versión tradicional del juego consiste en un tablero dividido en cuadrados con unas serpientes y unas escalerillas que conectan los cuadrados de diferentes filas. Generalmente es jugado por más de un jugador, y el objeto del juego es ser el primero en llegar, moviendo una ficha, desde el cuadro de la esquina inferior izquierda hasta el de la esquina superior derecha. Los jugadores mueven alternativamente, y el número de cuadros a mover está indicado por el lanzamiento de un dado. Si un movimiento lleva el jugador hasta la cabeza de una serpiente, entonces éste retrocede hasta la posición ocupada por la cola. Del mismo modo, si el jugador cae a los pies de una escalerilla, sube por ella hasta el cuadro en que está el último escalón. En resumen, al caer en la cabeza de una serpiente se retrocede, y al caer en los pies de una escalerilla se avanza.

En la versión para ordenador del juego, sería más interesante crear una distribución aleatoria de serpientes y escalerillas que diera lugar a un tablero diferente cada vez que se juega. La posición actual de cada jugador se puede señalar por medio de un carácter gráfico con forma de hombre, lo cual sugiere varias posibilidades de animarlo, deslizándose por las serpientes y subiendo por las escaleras. Otra de las cosas que debe hacer el programa es llevar la cuenta de los turnos de movimiento de los jugadores, e ¡incluso generar números aleatorios para no necesitar un dado!

Como ya se ha mencionado, el principal reto de este juego, es diseñar y manejar los gráficos que imprimen el tablero, y cada una de las serpientes y escalerillas. Un ejemplo de tablero se puede ver en la figura 7. 1. Las serpientes y las escalerillas se han producido utilizando varios caracteres gráficos combinados.

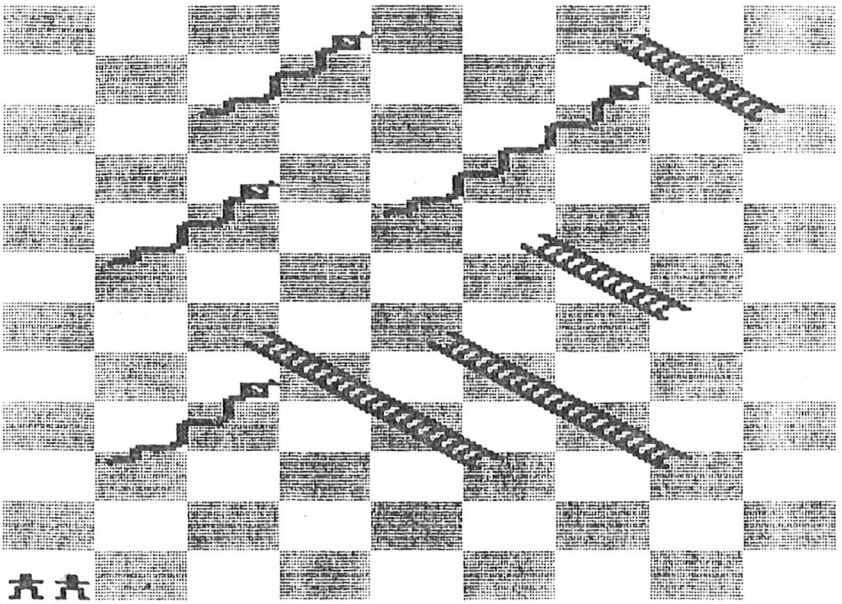


Fig. 7. 1

No sólo son aleatorias las posiciones de las serpientes y las escalerillas, también lo son las longitudes de las mismas.

El único problema real para contruir el tablero es asegurarse de que las serpientes y las escalerillas no «chocan», es decir no cruzan unas por encima de las otras. Hay dos formas de asegurar esta separación. La primera es la siguiente: la posición del principio y el final de cada serpiente y cada escalerilla se puede almacenar en una matriz, y cada vez que algo nuevo se va a añadir al tablero, se examina la matriz para ver si solapa con algo que ya estaba impreso. La segunda consiste simplemente en examinar las posiciones en las que se va a imprimir lo nuevo, para asegurarse de que hay un carácter en blanco. En la práctica, a menos que la velocidad sea muy importante, es muy fácil ver qué es lo que ya está impreso en la pantalla (por supuesto, aunque el método sea simple en teoría, hay que solventar varias dificultades prácticas, que se describen en secciones posteriores). Este mismo método de comprobación se puede emplear para descubrir si un jugador aterriza en la cabeza de una serpiente, o a los pies de una escalerilla. De hecho, se podría decir que

en este juego todo se basa en encontrar qué carácter está ya impreso en la pantalla.

El Programa principal

Aunque éste no sea un juego de gráficos dinámicos, su programa principal tiene una forma conocida:

```
 1Ø REM serpientes y escaleras
 3Ø GO SUB 1ØØØ
 4Ø GO SUB 8ØØØ
 6Ø GO SUB 25ØØ
 7Ø GO SUB 37ØØ
 8Ø REM bucle
 9Ø LET I=1: GO SUB 5ØØØ
1ØØ IF NOT HOME THEN  LET I=2:
      GO SUB 5ØØØ
11Ø IF NOT HOME THEN  GO TO 8Ø
12Ø GO SUB 95ØØ
13Ø IF AGAIN THEN  RUN
14Ø STOP
```

La línea 30 llama a la subrutina de inicialización que está en la línea 1000. La subrutina del título que empieza en la 8000 es llamada por la línea 40. Después la línea 60 llama a la subrutina 2500, que imprime el tablero en la pantalla. Luego la subrutina 3700 es llamada para inicializar algunas variables antes de empezar el juego. El bucle, líneas 80 a 110, es el responsable de ejecutar el juego. Ambos jugadores mueven una vez dentro del bucle. La línea 90 llama a la subrutina 5000 para permitir a un jugador un movimiento, y la línea 100 llama a la misma subrutina para permitir un movimiento al otro jugador. La variable HOME se utiliza para indicar que uno de los jugadores ha alcanzado la esquina superior izquierda del tablero. Por fin, cuando el juego se ha terminado, la línea 120 llama a la subrutina 9500, y la línea 130 termina el juego, o vuelve a empezar dependiendo del valor de la variable AGAIN.

Subrutinas 1000 y 8000

La subrutina 1000, simplemente establece los caracteres gráficos y las matrices que se van a emplear:

```
1ØØØ REM inicio
1Ø1Ø DATA "A",Ø,2,127,1Ø2,116,
     124,252,192
1Ø2Ø DATA "B",3,3,3,3,3,7,254,
     252
```

```
1030 DATA "C",0,0,0,0,0,0,1,3
1040 DATA "D",3,3,31,62,32,0,0,0
1050 DATA "E",0,0,6,3,1,35,54,28
1060 DATA "F",152,60,102,207,
     217,115,54,28
1070 DATA "G",0,0,0,0,128,192,
     96,240
1080 DATA "H",13,7,3,1,0,0,0,0
1090 DATA "I",152,60,102,192,
     192,96,0,0
1100 DATA "J",24,24,126,24,60,
     36,36,102
1110 RESTORE 1000
1120 FOR I=1 TO 10
1130 READ C$
1140 FOR J=0 TO 7
1150 READ D
1160 POKE USR C$+J,D
1170 NEXT J
1180 NEXT I
1185 DIM V(2): DIM N$(2,8)
1186 DIM X(2): DIM Y(2)
1199 RETURN
```

La forma de unir los caracteres para hacer una serpiente o una escalerilla es un poco complicada, y se entiende mejor mirando la Figura 7. 2. Las matrices X e Y almacenan las coordenadas de la figura del hombre «J», para cada jugador. Es más fácil describir la utilización de la matriz V más tarde, al hablar del método para hacer cada movimiento.

La matriz alfanumérica N$ guarda los nombres de cada uno de los jugadores: N$ (1) guarda el nombre del jugador uno, y N$(2) guarda el nombre del jugador dos.

La subrutina 8000 imprime las instrucciones, pregunta sus nombres a los jugadores, y también si quieren un dado real o no.

```
8000 REM titulo
8010 PAPER 6
8020 BORDER 6: INK 7
8025 CLS
8030 GO SUB 8500
8035 PAPER 8
8040 PRINT AT 8,1;"Este es un ";
     "juego para "; INK 0;"dos"
8045 PRINT AT 9,1;"jugadores."
```

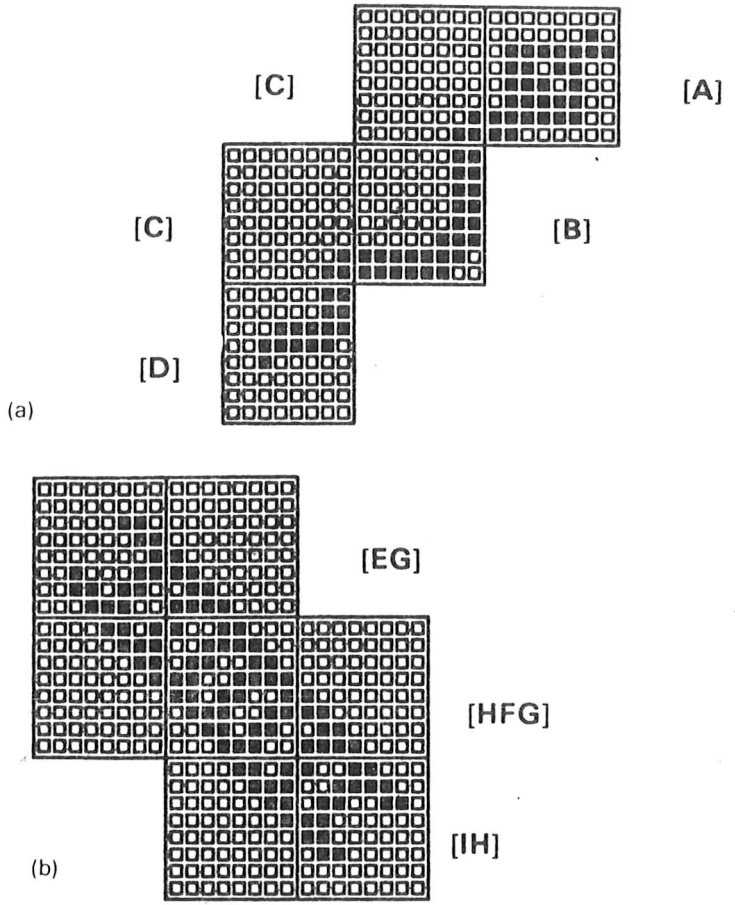


Fig. 7.2

Caracteres gráficos de (a) serpiente y (b) escalera.

```
8050 PRINT AT 10,1;"El primer ";
     "jugador que "; INK 0;
     "alcanze"
8060 PRINT AT 11,1;"la parte ";
     "superor izqui"; INK 0;
     "erda de"
8070 PRINT AT 12,1;
     "la pantalla gana"
8080 PRINT AT 21,1; PAPER 8;
      INK 0;"quieres usar un ";
      "dado de verdad?"
8090 INPUT A$
```

```
8100 IF A$(1)<>"S" AND
     A$(1)<>"N" THEN  GO TO 8090
8110 LET DICE=0
8120 IF A$(1)="S" THEN
      LET DICE=1
8150 INPUT "Nombre del primer ";
     "jugador ?";N$(1)
8160 INPUT "Nombre de otros ";
     "jugadores ?";N$(2)
8170 FOR I=1 TO 2
8180 FOR N=8 TO 2 STEP -1
8190 IF N$(I,N)=" " THEN
      LET N$(I)=" "+N$(I,1 TO 7)
8200 NEXT N
8210 NEXT I
8299 RETURN
```

Las líneas 8000 a 8120 establecen la variable DICE de acuerdo con la elección de los jugadores: DICE = 1, si quieren un dado real y DICE = 0, si quieren que el ordenador genere números aleatorios. Las líneas 8150 y 8160 toman los nombres de los jugadores, y los almacenan en N$(1), y N$(2). Las líneas 8170 y 8210 ajustan los nombres de los jugadores, para que los mensajes posteriores aparezcan más cuidados.

La subrutina 8000 llama a la subrutina 8500 para imprimir el título:

```
8500 REM s & l
8510 GO SUB 2000
8520 PAPER 8
8530 LET X1=1
8540 LET X2=16
8550 FOR Y=5 TO 2 STEP -1
8560 PRINT AT Y,X1;"[CB CB]";
     AT Y,X2;"[HFG HFG]"
8570 LET X1=X1+1
8580 LET X2=X2-1
8590 NEXT Y
8600 PRINT AT 1,5;"[CA CA]";
     AT 1,13;"[EG EG]"
8610 PRINT AT 6,1;"[D  D]";
     AT 6,17;"[HI  HI]"
8620 PRINT AT 4,10;" Y ";
8699 RETURN
```

Esta subrutina imprime dos serpientes y dos escalerillas sobre el tablero, con la palabra «Y» en medio.

Impresión del tablero —Subrutina 2500 y subrutinas asociadas

La subrutina 2500 imprime el tablero completo con serpientes y escalerillas. Lo hace por medio de otras dos subrutinas:

```
2500 REM serp. y esc.
2510 GO SUB 2000
2520 FOR S=1 TO 3+RND*2
2530 GO SUB 3000
2540 GO SUB 3500
2550 NEXT S
2599 RETURN
```

La subrutina 2000 imprime el patrón de cuadros negros y rojos, que configuran el tablero. Las subrutinas 3000 y 3500 imprimen una única serpiente, y una única escalerilla respectivamente. El bucle FOR, líneas 2520 a 2545, llama a las subrutinas 3000 y 3500 un número aleatorio de veces, entre tres y cinco. La subrutina 2000 es:

```
2000 REM cuadricula
2010 PAPER 6
2020 CLS
2025 LET C=0
2030 FOR Y=1 TO 19 STEP 2
2040 FOR X=1 TO 21 STEP 2
2045 PAPER C
2046 LET C=2*(NOT C)
2050 PRINT AT Y,X;"  ";
2060 PRINT AT Y+1,X;"  ";
2070 NEXT X
2080 NEXT Y
2099 RETURN
```

Cada uno de los cuadros que forman el tablero está compuesto de cuatro blancos, que se imprimen alternativamente en negro y rojo, utilizando la variable C, para asignar el color (línea 2045).

La subrutina 3000 parece un poco complicada, pero es sólo por el gran número de sentencias PRINT empleadas para hacer la serpiente:

```
3000 REM hacer serpiente
3002 INK 7
3005 LET TRY=0
3010 LET X=INT (RND*6)+2:
      LET Y=INT (RND*6)+1
3012 LET L=INT (RND*(8-Y))+3
```

```
3Ø15 IF X-1<(8-Y) THEN
      LET L=INT (RND*(X-2))+3
3Ø16 IF TRY>2Ø THEN  RETURN
3Ø18 LET TRY=TRY+1
3Ø2Ø LET F=-1: GO SUB 9ØØØ:
      IF CLASH THEN  GO TO 3Ø1Ø
3Ø4Ø PRINT AT Y*2,X*2-1;
      PAPER 8;"[CA]";
3Ø5Ø FOR Z=X-1 TO X-L+2 STEP -1
3Ø55 LET Y=Y+1
3Ø6Ø PRINT AT Y*2-1,Z*2;
      PAPER 8;"[CB]";
3Ø7Ø PRINT AT Y*2,Z*2-1;
      PAPER 8;"[CB]";
3Ø8Ø NEXT Z
3Ø85 LET Y=Y+1
3Ø9Ø PRINT AT Y*2-1,Z*2+1;
      PAPER 8;"[D]";
3Ø99 RETURN
```

La línea 3010 genera dos números aleatorios que especifican la posición inicial de una serpiente. La línea 3012 genera otro número aleatorio que asigna a L, y que determina la longitud de la serpiente. La sentencia IF asegura que L no hace salir a la serpiente de la pantalla. La línea 3020 utiliza la subrutina 9000 para comprobar que la serpiente definida por X, Y y L no se cruza con otra serpiente o con una escalerilla. Si es así, el control pasa a la línea 3010 para generar otro conjunto de valores X, Y y L que se vuelven a comprobar. Así probamos varias posiciones hasta encontrar una que sea adecuada, que no se solape con cosas ya impresas en el tablero. Como no tenemos garantizado que el tiempo empleado sea razonable, la variable TRY se usa para contar el número de intentos. La línea 3016 sólo nos deja probar 20 veces.

Una vez encontrada ya una serpiente adecuada, las líneas 3040 a 3090 imprimen la combinación de caracteres que la hacen aparecer por fin en la pantalla. Para igualar con el color del carácter en dicha posición, utilizamos PAPER 8, así, si la serpiente está sobre un cuadro rojo, el fondo será negro.

La subrutina 3500 funciona como la 3000 aproximadamente:

```
35ØØ REM hacer escalera
35Ø5 LET TRY=Ø
351Ø LET X=INT (RND*7)+1:
      LET Y=INT (RND*8)+1
3515 IF TRY>2Ø THEN  RETURN
3516 LET TRY=TRY+1
3517 LET L=INT (RND*(8-Y))+2
```

```
3520 IF 8-X<9-Y THEN
      LET L=INT (RND*(6-X))+3
3530 LET F=1: GO SUB 9000:
      IF CLASH THEN  GO TO 3510
3540 PRINT AT Y*2,X*2; PAPER 8;
     "[EG]";
3545 LET Y=Y+1
3550 FOR Z=X TO X+L-2
3560 PRINT AT Y*2-1,Z*2;
      PAPER 8;"[HFG]";
3570 PRINT AT Y*2,Z*2+1;
      PAPER 8;"[HFG]";
3580 LET Y=Y+1
3590 NEXT Z
3600 PRINT AT Y*2-1,Z*2;
      PAPER 8;"[HI]";
3699 RETURN
```

Las líneas 3505 a 3530 encuentran valores para X, Y y L para producir una escalerilla que no interfiera con otras serpientes o escalerillas ya impresas en la pantalla. Las líneas 3540 a 3600 imprimen la combinación de caracteres que producen por fin la escalerilla.

La subrutina 9000 examina lo que está ya impreso en la pantalla en cada una de las posiciones de carácter que van a ser afectadas al imprimir una escalerilla o una serpiente:

```
9000 REM choque
9006 LET CLASH=0
9007 LET T=0
9008 LET E=2*(X+F*L+F):
      LET U=X*2-F: LET V=Y*2-1
9010 REM bucle
9020 IF SCREEN$ (V,U)<>" " THEN
      LET T=1
9030 IF SCREEN$ (V,U+1)<>" "
      THEN  LET T=1
9040 IF SCREEN$ (V,U-1)<>" "
      THEN  LET T=1
9050 LET V=V+1: LET U=U+F
9060 IF U<>E AND T<>1 THEN
      GO TO 9010
9070 LET CLASH=T
9099 RETURN
```

La línea 9008 indica qué posiciones tienen que ser examinadas. Entonces las líneas 9010 a 9060 proceden a hacerlo, examinando cada po-

sición en turno hasta que se llega a la última o se encuentra un carácter que no está en blanco. Las líneas 9020 a 9040 hacen la comprobación de lo ya pintado en cada posición utilizando la función SCREEN$. SCREEN$(X,Y), es una función que, en general, tiene como resultado el carácter impreso en X, Y. Sin embargo, en el caso de los caracteres gráficos definidos por el usuario, SCREEN$ devuelve la cadena vacía. Por tanto, para comprobar por medio de SCREEN$ que el carácter no es igual a espacio, la presencia de un gráfico de usuario (el resultado es la cadena vacía) puede ser detectada.

Subrutina 3700

La subrutina 3700 inicializa las matrices que más tarde se utilizarán, e imprime las dos figuras de hombre en sus posiciones iniciales:

```
3700 REM arranque
3710 FOR I=1 TO 2
3720 LET X(I)=1: LET Y(I)=10
3730 LET V(I)=+1
3735 INK 5+I
3736 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8;"[J]";
3740 NEXT I
3750 LET HOME=0
3760 LET FLAG=0
3799 RETURN
```

Como ya hemos dicho, X e Y sirven para guardar las posiciones de los caracteres del hombre de los dos jugadores. Aunque cada cuadro del tablero está formado por cuatro espacios de carácter, como las posiciones están almacenadas, se tiene que X(I)*2; Y(I)*2 es el carácter de la esquina inferior izquierda del cuadro de la X (I) —ésima columna e Y(I) —ésima fila del cuadrado. La matriz V almacena la dirección de movimiento de cada hombre. En cada actualización, la posición del hombre del jugador I—ésimo sufre los cambios:

```
LET X(I)=X(I)+V(I)
LET Y(I)=Y(I)-1
```

Hombre en movimiento —Subrutina 5000 y subrutinas asociadas

Una vez dibujado el tablero, y colocados los hombres en sus posiciones iniciales, la subrutina 5000 es llamada repetidamente para mover los hombres sobre el tablero:

```
5000 REM tirada
5010 IF DICE THEN  GO SUB 5500:
       GO TO 5020
5015 GO SUB 5600
5020 REM bucle
5030 GO SUB 4000
5035 BEEP .1,5+I*4
5040 LET M=M-1
5050 LET HOME=(X(I)=1 AND
     Y(I)=1)
5060 IF NOT HOME AND M THEN
      GO TO 5020
5065 IF SNAKE AND LADDER THEN
      GO SUB 5200
5070 IF SNAKE THEN  LET UPD=1:
       GO SUB 6000
5080 IF LADDER THEN  LET UPD=-1:
       GO SUB 6000
5090 LET HOME=(X(I)=1 AND
     Y(I)=1)
5099 RETURN
```

Dependiendo del valor de DICE, las líneas 5010 y 5015 llaman a una de las dos subrutinas del dado, bien para pedir al jugador I el resultado del lanzamiento de un dado real (subrutina 5500), o bien para generar un dado en la pantalla, que es lanzado para conocer el siguiente movimiento del jugador (subrutina 5600). Independientemente de la subrutina llamada, el resultado es devuelto a M, y las líneas 5030 a 5060 mueven el hombre I M cuadros. El movimiento de una sola casilla se produce llamando a la subrutina 4000 (línea 5030), y la línea 5050 comprueba si el movimiento va a llevar al hombre a la casilla superior izquierda. Después de cada llamada, la subrutina 4000 asigna 1 a SNAKE si el hombre ha aterrizado en la cabeza de una serpiente, y asigna 1 a LADDER si ha llegado a los pies de una escalera. Las líneas 5065 a 5080 comprueban si el hombre ha aterrizado en una serpiente o en una escalera, o en ambas a la vez. La línea 5065 resuelve el problema de la llegada a un cuadro que contiene la cabeza de una serpiente «y» el pie de una escalera llamando a la subrutina 5200 que elige «o» la serpiente «o» la escalera aleatoriamente. La subrutina 6000 llamada por las líneas 5070 Y 5080 se ocupa de hacer al hombre bajar por la serpiente, o subir por la escalera.

La subrutina 4000 tiene tres tareas diferentes: tiene que borrar el carácter del hombre de su antigua posición, actualizar las coordenadas teniendo en cuenta lo que sucede cuando el hombre llega al borde del tablero, y controlar la presencia de una cabeza de serpiente o un pie de escalera en cada nueva casilla.

```
4000 REM un movimiento
4006 INK 8
4010 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;"[J]";
4020 LET X(I)=X(I)+V(I)
4030 IF X(I)>11 THEN
      LET X(I)=11:
       LET Y(I)=Y(I)-1:
        LET V(I)=-V(I)
4040 IF X(I)<1 THEN  LET X(I)=1:
       LET Y(I)=Y(I)-1:
        LET V(I)=-V(I)
4045 LET SNAKE=0: LET LADDER=0
4046 GO SUB 4200
4050 IF SCREEN$ (Y(I)*2,
     X(I)*2)=" " THEN
      LET SNAKE=1
4060 IF SCREEN$ (Y(I)*2-1,
     X(I)*2-1)="(" THEN
      LET LADDER=1
4065 GO SUB 4400
4070 INK 6+I
4080 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;"[J]";
4099 RETURN


4200 REM poner pantalla
4210 LET SC1=PEEK 23606+
     256*PEEK 23607
4220 LET SC2=(PEEK 23675+
     256*PEEK 23676)-256
4230 POKE 23606,SC2-256*
     INT (SC2/256)
4240 POKE 23607,INT (SC2/256)
4299 RETURN

4400 REM reponer pantalla
4410 POKE 23606,SC1-256*
     INT (SC1/256)
4420 POKE 23607,INT (SC1/256)
4499 RETURN
```

El borrado del carácter del hombre de su antigua posición corre a cargo de la línea 4010 que reimprime el carácter del hombre en su antigua posición utilizando «OVER», y así se borra el antiguo hombre. La

actualización de la posición del hombre es llevada a cabo por las líneas 4020 a 4040. Las líneas 4030 a 4040 dan la impresión de ser una especie de «bote» en el borde del tablero. Cuando el hombre llega al borde del tablero, sube una fila (es decir Y(I) = Y(I)-1) y además invierte su dirección de movimiento (VX(I) = -VX(I)). Por último, las líneas 4045 a 4065 controlan la llegada a cabeza de serpiente y pie de escalera respectivamente, y la línea 4080 imprime el hombre en su nueva posición.

Las líneas 4050 y 4060 emplean la función SCREEN$ del Spectrum para buscar los caracteres de la cabeza de serpiente y pie de escalerilla. Desgraciadamente, como ya hemos mencionado antes, la función SCREEN$ no reconoce los caracteres gráficos definidos por el usuario, por tanto, la subrutina 4200 altera los caracteres que SCREEN$ controla, y así los caracteres gráficos sí serán reconocidos. Una vez realizada la comprobación, la subrutina 4400 vuelve las cosas a la normalidad. Ya sólo quedan por describir pequeñas subrutinas, y una más larga, la subrutina 6000. Antes de empezar con ella, veamos las pequeñas. La subrutina 5500 pregunta el resultado del lanzamiento de un dado real:

```
5500 REM dado1
5505 INK 2+I
5510 PRINT AT 21,0; PAPER 8;
     N$(I);" tira - Pulsa una ";
     "tecla"
5520 IF INKEY$="" THEN
      GO TO 5520
5530 INPUT "Que sacaste ? ";M
5531 BEEP .1,10
5540 IF M<1 OR M>6 THEN
      GO TO 5530
5599 RETURN
```

La subrutina 5600 utiliza la función RND, para simular un dado en la pantalla:

```
5600 REM dado2
5605 INK 6: PAPER 1+I
5606 LET THROW=RND*10+10
5610 FOR Q=18 TO 20
5620 PRINT AT Q,25;"    "
5630 NEXT Q
5632 PRINT AT 19,26; INK 6;
      PAPER 8;".";
5635 PAPER 8: INK 1+I
5636 LET M=1
5640 PRINT AT 21,0;N$(I);
     " tira - Pulsa una tecla"
5655 PAUSE 0
```

```
5656 PAPER 1+I
5660 FOR Q=1 TO THROW
5665 PAUSE Q+1
5666 INK 1+I
5670 GO SUB 5680+(M*20)
5673 LET M=INT (RND*6)+1
5674 INK 6
5675 BEEP .1,-15
5680 GO SUB 5680+(M*20)
5690 NEXT Q
5699 RETURN
5700 PRINT AT 19,26;".";
5710 RETURN
5720 PRINT AT 18,25;".";
     AT 20,27;".";
5730 RETURN
5740 GO SUB 5700
5750 GO SUB 5720
5759 RETURN
5760 GO SUB 5720
5770 PRINT AT 18,27;".";
     AT 20,25;".";
5779 RETURN
5780 GO SUB 5760
5790 GO SUB 5700
5799 RETURN
5800 GO SUB 5760
5810 PRINT AT 19,25;".";
     AT 19,27;".";
5819 RETURN
```

La subrutina anterior, dibuja un dado pequeño en la parte derecha de la pantalla (líneas 5610 a 5630). Después, dependiendo del valor aleatorio guardado en THROW, dado en la línea 5606, el dado rueda parándose suavemente mientras el bucle se ejecuta. El valor final se muestra cuando el bucle llega a la línea 5673 al acabar el tiempo.

La subrutina 5200 asigna 1 a SNAKE, ó 0 a LADDER, aleatoriamente para elegir una de las dos posibilidades:

```
5200 REM eleccion
5210 IF RND>.5 THEN
      LET LADDER=0: GO TO 5299
5220 LET SNAKE=0
5299 RETURN
```

La última subrutina de movimiento del hombre es la 6000. Es la res-

ponsable de que el hombre se deslice por una serpiente o suba por una escalera:

```
6000 REM arriba abajo
6010 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;"[J]";
6020 LET U=X(I)*2: LET V=Y(I)*2
6030 INK 8
6040 GO SUB 4400: PRINT AT V,U;
      PAPER 8; OVER 1;"[J]";
6045 IF FLAG=1 THEN  GO SUB 6500
6050 BEEP .1,20-2*Y
6060 PRINT AT V,U; PAPER 8;
      OVER 1;"[J]";
6070 LET V=V+UPD: LET U=U-1
6075 GO SUB 4200
6078 LET Q$=SCREEN$ (V,U)
6079 IF Q$="" THEN  GO SUB 6500:
       GO TO 6078
6080 IF (Q$<>"$" AND Q$<>"£")
      THEN  GO TO 6040
6085 IF FLAG=1 THEN  GO SUB 6500
6090 GO SUB 4400
6100 LET X(I)=INT (U/2)
6110 LET Y(I)=INT (V/2)
6120 IF UPD=1 THEN
      LET Y(I)=Y(I)+1:
       LET X(I)=X(I)+1
6130 INK 8
6140 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;"[J]";
6150 LET V(I)=-1
6160 IF Y(I)=INT (Y(I)/2)*2 THEN
      LET V(I)=1
6199 RETURN


6500 REM hombre
6510 PRINT AT V,U; PAPER 8;
      OVER 1;"[J]";
6520 IF FLAG=0 THEN  GO TO 6550
6530 LET FLAG=0
6540 GO TO 6599
6550 LET FLAG=1
6599 RETURN
```

La misión de la subrutina 6000 es simplemente imprimir el carácter del hombre en cada posición de la serpiente o la escalera, utilizando PRINT OVER para borrar la antigua versión del hombre (líneas 6010, 6040 y 6060), hacer los efectos sonoros adecuados (línea 6050), hasta que encuentra el carácter que marca el final de la serpiente o la escalera (línea 6080). El valor de UPD controla si el hombre ha de subir por una escalerilla (UPD = 1), o bajar por una serpiente (UPD = -1). Las subrutinas 4200 y 4400 vuelven a ser llamadas, y así SCREEN$ puede buscar los caracteres definidos por el usuario.

La subrutina 6500 se ocupa del hombre que está en el final de una escalera, o en el cuerpo de una serpiente o una escalera. Su forma de actuar es por medio de PRINT OVER para borrar el hombre, entonces se vuelve a usar SCREEN$ en la subrutina 6000 para comprobar la posición debajo de la cual estaba el hombre, y sigue la subrutina 6500 que es llamada otra vez para PRINT OVER el hombre nuevamente, y así reimprime el hombre en su anterior posición.

Con esto se acaba la descripción de parte lógica del movimiento de serpientes y escaleras, y hemos de reconocer que en gran medida ha sido desarrollada como resultado de los problemas surgidos al comprobarla. Por ejemplo, la situación en la que un hombre está subiendo por una escalera, en el final de la cual está el otro hombre, no fue prevista en los primeros planes del programa, y la subrutina 6500 tuvo que ser añadida. Es casi imposible tener en cuenta todas las cosas raras que pueden suceder antes de escribir un programa, pero, si se emplea una estructura modular, es muy fácil añadir nuevas subrutinas que hagan frente a dichos problemas.

La sección final del juego

La subrutina 9500 simplemente felicita al jugador que ha llegado primero, y pregunta si se quiere volver a jugar:

```
9500 REM fin del juego
9510 INK 1+I: PAPER 6+I
9520 PRINT AT 21,0; INK 0;N$(I);
     " gana"
9530 INPUT "Otro juego ?";A$
9540 IF A$(1)<>"S" AND
     A$(1)<>"N" THEN  GO TO 9530
9550 LET AGAIN=0
9560 IF A$="S" THEN  LET AGAIN=1
9599 RETURN
```

Evaluación

El programa final es rápido, y tiene unos gráficos buenos y efecti-

vos, pero está lleno de posibilidades de mejora, y también de memoria libre. Por ejemplo, si un jugador saca un seis, puede tirar el dado otra vez, o puede ser necesario un seis para empezar a mover. El principal problema de este juego, es que el resultado sólo depende de la suerte, y, aunque es bastante divertido, no es ningún reto, ni requiere ninguna destreza. Este problema sólo se puede resolver cambiando un poco la naturaleza del juego. La parte final del capítulo da las modificaciones necesarias para un nuevo juego. Mientras tanto, la versión electrónica de «Serpientes y escaleras» está lista para ser jugada.

La versión final —Listado completo

```
  10 REM serpientes y escaleras
  30 GO SUB 1000
  40 GO SUB 8000
  60 GO SUB 2500
  70 GO SUB 3700
  80 REM bucle
  90 LET I=1: GO SUB 5000
 100 IF NOT HOME THEN  LET I=2:
      GO SUB 5000
 110 IF NOT HOME THEN  GO TO 80
 120 GO SUB 9500
 130 IF AGAIN THEN  RUN
 140 STOP


 408 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;"[J]";
1000 REM inicio
1010 DATA "A",0,2,127,102,116,
     124,252,192
1020 DATA "B",3,3,3,3,3,7,254,
     252
1030 DATA "C",0,0,0,0,0,0,1,3
1040 DATA "D",3,3,31,62,32,0,0,0
1050 DATA "E",0,0,6,3,1,35,54,28
1060 DATA "F",152,60,102,207,
     217,115,54,28
1070 DATA "G",0,0,0,0,128,192,
     96,240
1080 DATA "H",13,7,3,1,0,0,0,0
1090 DATA "I",152,60,102,192,
     192,96,0,0
```

```
1100 DATA "J",24,24,126,24,60,
     36,36,102
1110 RESTORE 1000
1120 FOR I=1 TO 10
1130 READ C$
1140 FOR J=0 TO 7
1150 READ D
1160 POKE USR C$+J,D
1170 NEXT J
1180 NEXT I
1185 DIM V(2): DIM N$(2,8)
1186 DIM X(2): DIM Y(2)
1199 RETURN


2000 REM cuadricula
2010 PAPER 6
2020 CLS
2025 LET C=0
2030 FOR Y=1 TO 19 STEP 2
2040 FOR X=1 TO 21 STEP 2
2045 PAPER C
2046 LET C=2*(NOT C)
2050 PRINT AT Y,X;"  ";
2060 PRINT AT Y+1,X;"  ";
2070 NEXT X
2080 NEXT Y
2099 RETURN


2500 REM serp. y esc.
2510 GO SUB 2000
2520 FOR S=1 TO 3+RND*2
2530 GO SUB 3000
2540 GO SUB 3500
2550 NEXT S
3000 REM hacer serpiente
3002 INK 7
3005 LET TRY=0
3010 LET X=INT (RND*6)+2:
      LET Y=INT (RND*6)+1
3012 LET L=INT (RND*(8-Y))+3
3015 IF X-1<(8-Y) THEN
      LET L=INT (RND*(X-2))+3
3016 IF TRY>20 THEN  RETURN
```

```
3018 LET TRY=TRY+1
3020 LET F=-1: GO SUB 9000:
      IF CLASH THEN  GO TO 3010
3040 PRINT AT Y*2,X*2-1;
      PAPER 8;"  ";
3050 FOR Z=X-1 TO X-L+2 STEP -1
3055 LET Y=Y+1
3060 PRINT AT Y*2-1,Z*2;
      PAPER 8;"  ";
3070 PRINT AT Y*2,Z*2-1;
      PAPER 8;"  ";
3080 NEXT Z
3085 LET Y=Y+1
3090 PRINT AT Y*2-1,Z*2+1;
      PAPER 8;" ";
3099 RETURN


3500 REM hacer escalera
3505 LET TRY=0
3510 LET X=INT (RND*7)+1:
      LET Y=INT (RND*8)+1
3515 IF TRY>20 THEN  RETURN
3516 LET TRY=TRY+1
3517 LET L=INT (RND*(8-Y))+2
3520 IF 8-X<9-Y THEN
      LET L=INT (RND*(6-X))+3
3530 LET F=1: GO SUB 9000:
      IF CLASH THEN  GO TO 3510
3540 PRINT AT Y*2,X*2; PAPER 8;
     "  ";
3545 LET Y=Y+1
3550 FOR Z=X TO X+L-2
3560 PRINT AT Y*2-1,Z*2;
      PAPER 8;"   ";
3570 PRINT AT Y*2,Z*2+1;
      PAPER 8;"   ";
3580 LET Y=Y+1
3590 NEXT Z
3600 PRINT AT Y*2-1,Z*2;
      PAPER 8;"  ";
3699 RETURN


3700 REM arranque
```

```
3710 FOR I=1 TO 2
3720 LET X(I)=1: LET Y(I)=10
3730 LET V(I)=+1
3735 INK 5+I
3736 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8;" ";
3740 NEXT I
3750 LET HOME=0
3760 LET FLAG=0
3799 RETURN

4000 REM un movimiento
4006 INK 8
4010 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;" ";
4020 LET X(I)=X(I)+V(I)
4030 IF X(I)>11 THEN
      LET X(I)=11:
       LET Y(I)=Y(I)-1:
        LET V(I)=-V(I)
4040 IF X(I)<1 THEN  LET X(I)=1:
       LET Y(I)=Y(I)-1:
        LET V(I)=-V(I)
4045 LET SNAKE=0: LET LADDER=0
4046 GO SUB 4200
4050 IF SCREEN$ (Y(I)*2,
     X(I)*2)=" " THEN
      LET SNAKE=1
4060 IF SCREEN$ (Y(I)*2-1,
     X(I)*2-1)="(" THEN
      LET LADDER=1
4065 GO SUB 4400
4070 INK 6+I
4080 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;" ";
4099 RETURN


4200 REM poner pantalla
4210 LET SC1=PEEK 23606+
     256*PEEK 23607
4220 LET SC2=(PEEK 23675+
     256*PEEK 23676)-256
4230 POKE 23606,SC2-256*
     INT (SC2/256)
```

```
4240 POKE 23607,INT (SC2/256)
4299 RETURN


4400 REM reponer pantalla
4410 POKE 23606,SC1-256*
     INT (SC1/256)
4420 POKE 23607,INT (SC1/256)
4499 RETURN

5000 REM tirada
5010 IF DICE THEN  GO SUB 5500:
       GO TO 5020
5015 GO SUB 5600
5020 REM bucle
5030 GO SUB 4000
5035 BEEP .1,5+I*4
5040 LET M=M-1
5050 LET HOME=(X(I)=1 AND
     Y(I)=1)
5060 IF NOT HOME AND M THEN
      GO TO 5020
5065 IF SNAKE AND LADDER THEN
      GO SUB 5200
5070 IF SNAKE THEN  LET UPD=1:
       GO SUB 6000
5080 IF LADDER THEN  LET UPD=-1:
       GO SUB 6000
5090 LET HOME=(X(I)=1 AND
     Y(I)=1)
5099 RETURN


5200 REM eleccion
5210 IF RND>.5 THEN
      LET LADDER=0: GO TO 5299
5220 LET SNAKE=0
5299 RETURN


5500 REM dado1
5505 INK 2+I
5510 PRINT AT 21,0; PAPER 8;
     N$(I);" tira - Pulsa una ";
     "tecla"
```

```
5520 IF INKEY$="" THEN
      GO TO 5520
5530 INPUT "Que sacaste ? ";M
5531 BEEP .1,10
5540 IF M<1 OR M>6 THEN
      GO TO 5530
5599 RETURN


5600 REM dado2
5605 INK 6: PAPER 1+I
5606 LET THROW=RND*10+10
5610 FOR Q=18 TO 20
5620 PRINT AT Q,25;"    "
5630 NEXT Q
5632 PRINT AT 19,26; INK 6;
      PAPER 8;".";
5635 PAPER 8: INK 1+I
5636 LET M=1
5640 PRINT AT 21,0;N$(I);
     " tira - Pulsa una tecla"
5655 PAUSE 0
5656 PAPER 1+I
5660 FOR Q=1 TO THROW
5665 PAUSE Q+1
5666 INK 1+I
5670 GO SUB 5680+(M*20)
5673 LET M=INT (RND*6)+1
5674 INK 6
5675 BEEP .1,-15
5680 GO SUB 5680+(M*20)
5690 NEXT Q
5699 RETURN
5700 PRINT AT 19,26;".";
5710 RETURN
5720 PRINT AT 18,25;".";
     AT 20,27;".";
5730 RETURN
5740 GO SUB 5700
5750 GO SUB 5720
5759 RETURN
5760 GO SUB 5720
5770 PRINT AT 18,27;".";
     AT 20,25;".";
5779 RETURN
```

```
5780 GO SUB 5760
5790 GO SUB 5700
5799 RETURN
5800 GO SUB 5760
5810 PRINT AT 19,25;".";
     AT 19,27;".";
5819 RETURN

6000 REM arriba abajo
6010 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;" ";
6020 LET U=X(I)*2: LET V=Y(I)*2
6030 INK 8
6040 GO SUB 4400: PRINT AT V,U;
      PAPER 8; OVER 1;" ";
6045 IF FLAG=1 THEN  GO SUB 6500
6050 BEEP .1,20-2*Y
6060 PRINT AT V,U; PAPER 8;
      OVER 1;" ";
6070 LET V=V+UPD: LET U=U-1
6075 GO SUB 4200
6078 LET Q$=SCREEN$ (V,U)
6079 IF Q$="" THEN  GO SUB 6500:
       GO TO 6078
6080 IF (Q$<>"$" AND Q$<>"#")
      THEN  GO TO 6040
6085 IF FLAG=1 THEN  GO SUB 6500
6090 GO SUB 4400
6100 LET X(I)=INT (U/2)
6110 LET Y(I)=INT (V/2)
6120 IF UPD=1 THEN
      LET Y(I)=Y(I)+1:
       LET X(I)=X(I)+1
6130 INK 8
6140 PRINT AT Y(I)*2,X(I)*2-I+1;
      PAPER 8; OVER 1;" ";
6150 LET V(I)=-1
6160 IF Y(I)=INT (Y(I)/2)*2 THEN
      LET V(I)=1
6199 RETURN


6500 REM hombre
6510 PRINT AT V,U; PAPER 8;
      OVER 1;" ";
```

```
6520 IF FLAG=0 THEN  GO TO 6550
6530 LET FLAG=0
6540 GO TO 6599
6550 LET FLAG=1
6599 RETURN


8000 REM titulo
8010 PAPER 6
8020 BORDER 6: INK 7
8025 CLS
8030 GO SUB 8500
8035 PAPER 8
8040 PRINT AT 8,1;"Este es un ";
     "juego para "; INK 0;"dos"
8045 PRINT AT 9,1;"jugadores."
8050 PRINT AT 10,1;"El primer ";
     "jugador que "; INK 0;
     "alcanze"
8060 PRINT AT 11,1;"la parte ";
     "superor izqui"; INK 0;
     "erda de"
8070 PRINT AT 12,1;
     "la pantalla gana"
8080 PRINT AT 21,1; PAPER 8;
      INK 0;"quieres usar un ";
      "dado de verdad?"
8090 INPUT A$
8100 IF A$(1)<>"S" AND
     A$(1)<>"N" THEN  GO TO 8090
8110 LET DICE=0
8120 IF A$(1)="S" THEN
      LET DICE=1
8150 INPUT "Nombre del primer ";
     "jugador ?";N$(1)
8160 INPUT "Nombre de otros ";
     "jugadores ?";N$(2)
8170 FOR I=1 TO 2
8180 FOR N=8 TO 2 STEP -1
8190 IF N$(I,N)=" " THEN
      LET N$(I)=" "+N$(I,1 TO 7)
8200 NEXT N
8210 NEXT I
8299 RETURN
```

```
8500 REM s & l
8510 GO SUB 2000
8520 PAPER 8
8530 LET X1=1
8540 LET X2=16
8550 FOR Y=5 TO 2 STEP -1
8560 PRINT AT Y,X1;"      ";
     AT Y,X2;"        "
8570 LET X1=X1+1
8580 LET X2=X2-1
8590 NEXT Y
8600 PRINT AT 1,5;"      ";
     AT 1,13;"       "
8610 PRINT AT 6,1;"     ";
     AT 6,17;"       "
8620 PRINT AT 4,10;" Y ";
8699 RETURN

9000 REM choque
9006 LET CLASH=0
9007 LET T=0
9008 LET E=2*(X+F*L+F):
      LET U=X*2-F: LET V=Y*2-1
9010 REM bucle
9020 IF SCREEN$ (V,U)<>" " THEN
      LET T=1
9030 IF SCREEN$ (V,U+1)<>" "
      THEN  LET T=1
9040 IF SCREEN$ (V,U-1)<>" "
      THEN  LET T=1
9050 LET V=V+1: LET U=U+F
9060 IF U<>E AND T<>1 THEN
      GO TO 9010
9070 LET CLASH=T
9099 RETURN

9500 REM fin del juego
9510 INK 1+I: PAPER 6+I
9520 PRINT AT 21,0; INK 0;N$(I);
     " gana"
9530 INPUT "Otro juego ?";A$
9540 IF A$(1)<>"S" AND
     A$(1)<>"N" THEN  GO TO 9530
9550 LET AGAIN=0
9560 IF A$="S" THEN  LET AGAIN=1
9599 RETURN
```

Serpientes y escaleras de acción

La forma básica de Serpientes y escaleras permanece en la versión dinámica del juego. La idea es que en lugar de mover alternativamente cada hombre una cantidad aleatoria, un único hombre esté continuamente animado, moviéndose una casilla cada vez dentro del bucle de animación. Es claro que así el hombre automáticamente subiría todas las escaleras y bajaría por todas las serpientes que encontrara en su camino. Algunas veces la colocación de las serpientes y escaleras será tal, que el hombre llega al final, y otras el hombre se mueve continuamente en círculo, subiendo por las mismas escaleras, y bajando por las mismas serpientes. Para que resulte un juego de destreza, lo único necesario es añadir la condición extra de que para que el hombre suba por la escalera el jugador tiene que pulsar «L», justo cuando el hombre llega a los pies de la misma, y para impedir que el hombre se deslice por la serpiente, el jugador tiene que pulsar «S», justamente cuando el hombre llega a la cabeza. Si se pulsa cualquier otra tecla en cualquier otro momento, el hombre se para, y como el fin del juego es llegar lo más rápido posible al cuadro superior izquierdo, ¡ésta no sería una buena forma de jugar!

Las modificaciones son bastante simples. El programa principal queda así:

```
 10 REM serpientes y escaleras
 30 GO SUB 1000
 40 GO SUB 8000
 60 GO SUB 2500
 70 GO SUB 3700
 80 REM bucle
 90 LET I=1: GO SUB 5000
100 IF NOT HOME THEN  LET I=2:
      GO SUB 5000
110 IF NOT HOME THEN  GO TO 80
120 GO SUB 9500
130 IF AGAIN THEN  RUN
140 STOP
```

La subrutina 3700 se modifica para imprimir un solo hombre, y pone el tiempo a cero. Esto implica el cambio de la línea 3710 y añadir la línea 3755:

```
3710 FOR I=1 TO 1

3755 GO SUB 3800

3800 REM tiempo cero
3810 POKE 23674,0
```

```
3820 POKE 23673,0
3830 POKE 23672,0
3840 LET TIME=0
3899 RETURN
```

Las modificacions y adiciones de la subrutina 5000 son suficientes como para dar el listado completo de la nueva versión de la misma:

```
5000 REM tirada
5010 GO SUB 5300
5030 GO SUB 4000
5035 BEEP .1,5+I*4
5050 LET HOME=(X(I)=1 AND
     Y(I)=1)
5065 IF SNAKE AND LADDER THEN
      GO SUB 5200
5066 IF NOT SNAKE AND NOT LADDER
      AND INKEY$<>"" THEN
       GO SUB 5300: GO TO 5066
5070 IF SNAKE AND INKEY$<>"S"
      THEN  LET UPD=1:
        GO SUB 6000
5080 IF LADDER AND INKEY$="L"
      THEN  LET UPD=-1:
        GO SUB 6000
5090 LET HOME=(X(I)=1 AND
     Y(I)=1)
5099 RETURN
```

Ahora, la subrutina 5000 sólo llama una vez a la subrutina 4000 cada vez que entra en el bucle de animación, y llama a la subrutina 5300 para imprimir el tiempo actual. El resto de las modificaciones son la adición de la línea 5066, que para la animación si se pulsa «L» o «S» y el hombre no está en la cabeza de una serpiente, ni al pie de una escalera, y el cambio de las líneas 5070 y 5080, que hacen al hombre deslizarse por la serpiente si no está pulsada «S», y sube por la escalera sólo si «L» está pulsada. La subrutina 5300 es una subrutina nueva muy corta, que imprime el tiempo en la parte superior de la pantalla:

```
5300 REM tiempo
5310 LET TIME=(65536*PEEK 23674+
     256*PEEK 23673+PEEK 23672)/
     50
5320 PRINT AT 0,15; INK 0;
      PAPER 6;"Tiempo=";
      INT (TIME)
5399 RETURN
```

También damos el listado completo de la subrutina 8000, porque tiene bastantes cambios:

```
8000 REM titulo
8010 PAPER 6
8020 BORDER 6: INK 7
8025 CLS
8030 GO SUB 8500
8040 PRINT AT 8,1;"Debes tratar"
     ;" de que tu "; INK 0;
     "hombre"
8045 PRINT AT 9,1;"alcanze la ";
     "parte super"; INK 0;"ior"
8050 PRINT AT 10,1;"izquierda ";
     "de la pantal"; INK 0;"la"
8060 PRINT AT 11,1;"lo antes ";
     "posible."
8070 PRINT AT 12,1;"Debes ";
     "evitar ";
8080 PRINT AT 13,1;"las cabezas"
     ;" de las"
8090 PRINT AT 14,1;"serpientes "
     ;"pulsando S"
8100 PRINT AT 15,1;"y puedes ";
     "trepar por la"; INK 0;"s"
8110 PRINT AT 16,1;"escaleras ";
     "pulsando L"
8120 PRINT AT 21,1; INK 0;
     "Pulsa una tecla para ";
     "empezar"
8130 IF INKEY$="" THEN
      GO TO 8130
8299 RETURN
```

Igualmente en la subrutina 9500:

```
9500 REM fin del juego
9510 INK 1+I: PAPER 6+I
9520 PRINT AT 21,0;"Tardaste ";
     INT (TIME)
9530 INPUT "Otro juego ?";A$
9540 IF A$(1)<>"S" AND
     A$(1)<>"N" THEN  GO TO 9530
9550 LET AGAIN=0
9560 IF A$="S" THEN  LET AGAIN=1
9599 RETURN
```

Con estos cambios del original, se consigue un juego más rápido y excitante. El hombre se mueve por el tablero, bajando serpientes y subiendo escaleras ¡de un modo fascinante! Se pueden conseguir muchas mejoras de este juego a partir de las características de los primeros. Por ejemplo, la sección final, puede publicar mensajes, se puede imponer el tiempo límite, hacer una tabla de los mejores tiempos, mejorar los efectos sonoros..... incluso introducir nuevos elementos como casillas «AGUJERO NEGRO», que hay que saltar, etc... La lección que hemos aprendido, es que se pueden hacer muchas cosas con los juegos tradicionales, para ponerlos al día, y que los resultados sean buenos.

8. Convertido en programador experto

El paso de programador «novato» a experto en la materia sólo requiere desarrollar todas las destrezas que se han ido adquiriendo, e ir ganando confianza en uno mismo, para aplicarlas a proyectos cada vez más ambiciosos. Si usted ha ido tecleando todos los programas de los capítulos anteriores, habrá ido comprendiendo algunas de las formas de escribir programas largos. De todos modos, para aprender, lo más esencial es ganar práctica, primero modificando y extendiendo los programas del libro, y más tarde, escribiendo programas de su «propia cosecha».

No se llega a ser un buen programador si no se escriben programas, para aprender de los éxitos y errores propios.

En este capítulo final daremos un vistazo a los métodos y actitudes con los que uno debe enfrentarse a los programas. El capítulo 1 hizo hincapié en la necesidad de utilizar un método de programación, y todos los capítulos de este libro se han escrito utilizando programación estructurada y refinamiento progresivo, pero esto no es todo. El lector podría pensar que estas herramientas configuran el «lado científico» de la tarea de programar, pero, si no fuera por estos métodos, programar sería muy difícil.

De todos modos, incluso utilizando dichos métodos, para llegar a ser un buen programador se necesita mucha práctica, porque hay muchos trucos que sólo se aprenden practicando.

Esto no quiere decir que la buena programación sea la programación «astuta», en el sentido del empleo de trucos y métodos ingeniosos. Un buen programa debe ser lo más claro posible, y dichos trucos y métodos ingeniosos tienden a aumentar la confusión del programa, y un programa confuso es terreno abonado para los errores. Por tanto, el único modo de llegar a ser un maestro en programación es aprender de la propia experiencia como escritor de programas, pero antes de nada hay que conocer qué cosas son útiles, y acabar el programa que se ha empezado.

Juegos y solución de problemas

Si volvemos sobre los primeros capítulos, veremos que casi todos los

programas partieron de una idea sencilla que, una vez escrita en BASIC, no produjo un juego demasiado satisfactorio. La verdad es que la mayor parte de las ideas para juegos parecen no estar muy de acuerdo con el primer programa que escribimos, pero ésta no es una razón para abandonar la idea. El éxito de cualquier juego de ordenador depende de la forma de escribirlo, e incluso grandes ideas mal escritas dan como resultado juegos horribles. Por ejemplo, uno de los juegos antiguos y que crean más adicción, «Invasores del espacio», puede resultar insoportable si no se presta atención a los detalles. Si el programa es demasiado lento o si la respuesta al teclado no es rápida, el juego puede resultar frustrante en lugar de divertido.

Si el juego no está de acuerdo con la idea, no hay que descartarla, y buscar algo nuevo, hay que hacerse la siguiente pregunta: ¿Qué falla?. Muchas veces la respuesta es fácil de encontrar y no es difícil corregir el fallo. Por ejemplo, si el juego es demasiado lento, el truco puede estar en escribir algunas subrutinas en código máquina. Por otra parte, la razón de una respuesta lenta del teclado puede no ser fácil de encontrar. Esto puede ser debido al tiempo necesario para ejecutar el bucle de animación, pero es posible que se deba al hecho de no inspeccionar durante un tiempo demasiado largo el teclado dentro del bucle.

Por ejemplo, si el bucle de animación es demasiado rápido, la mejor forma de frenarlo es utilizando un bucle de espera. Si el tiempo perdido en dicho bucle de espera es una parte significativa del tiempo empleado por el bucle de animación, el teclado será inspeccionado cada vez durante un tiempo muy corto. La solución no es aumentar la velocidad del bucle, sino examinar el teclado más de una vez en cada una de las ocasiones en que se ejecuta el bucle.

Algunas veces el primer intento para escribir un juego no tiene éxito, y no es por culpa de la forma de escribirlo. Por ejemplo, en el capítulo 2 la primera versión de «La colina de las hormigas» no fue tan divertida como se pensaba, porque resultaba demasiado fácil. Si un juego es demasiado fácil, se suele tender a complicarlo estableciendo un tiempo límite muy corto, pero a menos que el tiempo necesario realmente dependa de la habilidad del jugador, el acortar dicho tiempo límite, o bien no tendrá ningún efecto o bien hará el juego imposible. Lo que de verdad hay que hacer es examinar cuidadosamente el juego, y descubrir qué es lo que lo hace difícil.

En el caso de «La colina de las hormigas», el problema radicaba en que las hormigas no eran suficiente obstáculo para que el hombre alcanzase el nido. En este caso particular, la solución fue dispersar las hormigas de forma más aleatoria, y confinarlas a un área más pequeña. Si después de examinar el juego la causa de su facilidad o dificultad resulta ser algo imposible de corregir, entonces no queda elección, hay que dejarlo e intentar algo de nuevo, pero ésta es una situación muy poco frecuente.

Lo cierto es que muy pocos juegos o cualquier otro programa corto

de este tipo, son lo suficientemente fáciles como para que la primera versión escrita llene nuestras expectativas, y esto no es bastante razón como para abandonar y probar algo nuevo. Si de verdad se pretende programar mejor, es necesario esforzarse en conseguir transformar el programa que tenemos en el programa que de verdad queremos, y esto es cuestión de observar cuidadosamente el programa, e ir resolviendo cada uno de los pequeños problemas que se han planteado.

Añadiendo los toques finales

Una vez conseguido un juego divertido, el paso siguiente es añadir los procedimientos necesarios para hacerlo «jugable». Es decir, hay que comprobar si otras personas pueden jugar, y el único modo de comprobar si hemos tenido éxito es observar a otras personas mientras juegan. No hay duda de que otra persona puede fracasar el programa por culpa de un malentendido, y por tanto, la culpa será suya, pero, si hubo un malentendido, es probablemente porque el juego no está bien explicado o porque es demasiado complicado. Pero si el programa está bien hecho, en ningún caso debería fracasar.

Todos los juegos de este libro están «a prueba de golpes», en el sentido de que es imposible introducir valores o controlar el juego de modo que se interrumpa y devuelva al jugador al BASIC. El método mejor para hacer las comprobaciones está basado en la vieja y conocida idea de «Basura dentro basura fuera» (en inglés *garbage in garbaje out*, abreviado GIGO).

Si podemos evitar que las cosas inútiles entren en el programa, podremos estar más seguros de no fracasar.

Por ejemplo, si se pide al jugador que elija el nivel de dificultad, hemos de estar seguros que la respuesta está dentro de los límites correctos antes de seguir con el resto del programa. De cualquier modo, aunque se compruebe cada valor introducido, todavía es posible que el programa se interrumpa o no siga el recorrido previsto por culpa de una combinación de valores que no habíamos previsto. Por ejemplo, en las primeras pruebas de «serpientes y escalerillas» uno de los tableros generados incluía un cuadro en el que coincidían la cabeza de una serpiente y los pies de una escalera. El resultado era que el hombre tenía que subir por la escalera y bajar por la serpiente al mismo tiempo, con las obvias consecuencias desastrosas para el resto del programa. Hubo que insertar una línea para que el hombre eligiese una de las dos opciones.

Para asegurarnos de que un programa no falla por culpa de circunstancias inesperadas, hay que comprobarlo bien, pero no es lo mismo comprobarlo «bien» que jugarlo mucho tiempo. Si se va probando el juego a medida que se va desarrollando, cuando por fin se termina de escribir, se ha llegado a ser realmente hábil en el juego, y esto quiere decir que estamos probando el juego con un determinado nivel de dificul-

tad, y los problemas que podrían causar el fracaso sólo aparecen en un nivel de destreza más bajo. Por ejemplo, supongamos que en una prueba de la primera versión de «el salto de la rana» (capítulo 3) jugamos muchas veces con éxito, pero llega un jugador novato y en el primer juego el programa «casca», porque el jugador falla con las diez ranas y la puntuación es cero.

Cuando se comprueba un programa, hay que hacerlo de la manera más tonta posible, para intentar hacerlo fracasar. Si esto no se consigue por más que se intente, con un poco de suerte los errores que existan serán pequeños.

Hay otro problema que no se suele presentar en otros tipos de programas y es la aleatoriedad. Por ejemplo, el tablero de «Serpientes y escalerillas» se genera al azar, y es probable que haya que esperar mucho tiempo hasta que se genere un tablero problemático (es decir, uno que contenga una cabeza de serpiente y un pie de escalera en el mismo cuadro).

Hay dos formas de enfrentarse a la comprobación de la aleatoriedad de un programa. La primera consiste en cambiar cualquier sentencia que contenga RND por valores fijos que pueden causar problemas.

El problema de este método es que hay que pensar en los valores candidatos a causar problemas, y, si se puede pensar en ellos, es que son tan evidentes que ni siquiera hemos de ejecutar el programa para descubrirlos.

Por ejemplo, si sospechamos que hay valores aleatorios en el juego de «Serpientes y escalerillas» que imprimen la cabeza de la serpiente y el pie de la escalera en el mismo cuadro, es evidente que serán los responsables del problema.

El segundo método es simplemente observar muchos casos. Este ha sido uno de los métodos empleados para comprobar «Serpientes y escaleras» durante su desarrollo. Para poder ver muchos ejemplos en un intervalo razonable de tiempo, el programa principal ha sido cambiado por un bucle, que sólo llama a las subrutinas que pintan el tablero una y otra vez. Una vez localizado el problema, hay que asegurarse de que hace lo que pensábamos, antes de dar por localizado el error.

En este caso, una vez conseguido el tablero con la cabeza de serpiente y el pie de escalera en el mismo lugar, las funciones RND se reemplazan por constantes que produzcan el mismo efecto, y así, cada vez que el programa se ejecuta, se obtiene un tablero problemático. Esta es la única manera de probar la efectividad de fijar un error en un programa que emplea aletoriedad.

Imaginemos que después de unas horas de probar, aparece un problema y el programa «casca». Si se fija el error inmediatamente,

puede suceder que haya que esperar varias horas para ver si estaba bien localizado.

Por otra parte, si se encuentra una forma de modificar el programa de modo que el problema aparezca cada vez que ejecutemos el programa, podemos estar seguros de que habíamos encontrado el error y no tenemos que estar comprobando el programa durante horas.

Las dos áreas en que hay que desarrollar un mayor esfuerzo son la comprobación y depuración de los programas.

Sin embargo, producir programas «usuario—amistosos» es igual de importante y se necesita la misma habilidad. En algunos sentidos es la parte que requiere más habilidad, porque es casi imposible definir qué es lo que hace a un programa ser así. La única forma de descubrir qué cosas son útiles, es ver el modo en que otros usuarios reaccionan ante un programa.

Cuando se diseña un programa, hay que intentar pensar en la manera en que va a ser utilizado, y, de acuerdo con ella, encontrar el orden «natural» de ejecución del mismo.

Por tanto, si producimos un programa que nos resulte sencillo de usar, hemos de recordar que no somos el usuario típico. Los programadores tienden a olvidar lo difícil y confuso que resulta un teclado y una pantalla a un no-programador, y, además, el programador conoce su programa mejor que nadie. Así resulta que algo que para nosotros es obvio y fácil, parece oscuro y difícil a un no-programador.

Cualquier programador habrá tenido la experiencia de ver en otra persona utilizar su programa de una forma tan «patosa» que habrá llegado a exasperarse por culpa de tanta estupidez. Pero esta actitud es incorrecta, los usuarios siempre tienen razón, y, si encuentran el programa difícil de usar, hay que pensar que el problema está en el programa, ¡no en el usuario!

Posibles juegos

Si por fin el lector se ha inspirado y ha pensado en intentar algo por su cuenta, puede resultarle útil conocer algo acerca de los tipos de juegos que son más populares en la actualidad.

Los «juegos de marcianitos» o de animación (en inglés «juegos de arcada») forman el más amplio y popular grupo de juegos. Con la posible excepción de «Serpientes y escalerillas», todos los juegos están basados en la animación.

La principal característica de todos estos juegos es la forma en que el jugador controla el movimiento sobre la pantalla de algunos objetos.

Aunque sea capaz de crear nuevos juegos con nuevos temas y objetivos, lo único que será diferente es la forma en que el jugador controla el objeto en movimiento. Por ejemplo, usted puede hacer un juego nuevo de «La colina de las hormigas» simplemente cambiando las hormigas por fantasmas y el nido por un cofre con un tesoro, pero se jugará igual que antes. Sin embargo, «la serpiente» en el capítulo 5 es muy diferente de «la colina» en lo que respecta a la habilidad que el jugador necesita para controlar el objeto móvil.

Para crear un juego animado, no sólo hay que pensar en el tema, sino también en la forma de llevar el control del objeto en movimiento, y qué nuevos elementos de destreza se necesitan.

El segundo grupo en popularidad son los «juegos de aventuras». Algunas veces constan de partes de animación simple, pero sólo como parte del objetivo final del juego. En un juego de aventuras que no se apoya en la animación para probar la destreza del jugador, se crea, en cambio, un mundo ficticio habitado por varias criaturas, algunas de las cuales son amigas y otras hostiles. El mundo y las criaturas se crean, generalmente, utilizando descripciones y algunos gráficos, y el jugador suele actuar sobre dicho mundo tecleando ciertas instrucciones. Estos juegos suelen presentar menos problemas de programación que los juegos de animación, pero necesitan mucha más imaginación.

Una tercera categoría son los «juegos de estrategia». Se incluyen aquí los de ajedrez, damas, tres en raya.... Los juegos de estrategia podrán, a veces, incluir gráficos complicados, (por ejemplo, imprimir un tablero de ajedrez con todas las piezas), pero la mayor dificultad suele ser conseguir que el ordenador ejecute un juego convincente. Conseguir, por ejemplo, que el ordenador juege al ajedrez nos lleva a las fronteras de la ciencia de la computación.

Cuando se está trabajando en un juego, siempre existe la posibilidad del descubrimiento sorpresa. Es divertido comprobar que al escribir un programa de juego, éste suele sugerir otros usos alternativos de los mismos elementos en formas diferentes. Esta es otra de las razones para escribir los programas empleando subrutinas.

En efecto, si un programa se ha escrito empleando subrutinas, éstas se podrán utilizar probablemente para construir otros programas, y, en este sentido, cuántos más juegos se escriban, más sencillo resultará escribir otros.

El camino hacia adelante

Si usted «ha jugado» con los juegos de este libro, en el sentido de alterarlos, y, en general, hacer pruebas con ellos, el siguiente paso consiste en escribir programas de su «propia cosecha», de modo que puedan ser

utilizados por otras personas, y resulten divertidos, en lugar de dejarlos a medias, de modo que sólo el que los ha escrito los comprenda.

Si lo consigue, se beneficiará doblemente del programa, una vez por divertirse jugando, y otra por la satisfacción de haber terminado un juego al que otros pueden jugar.